US 20070016405A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2007/0016405 A1

Mehrotra et al. (43) Pub. Date: Jan. 18, 2007

(54) **CODING WITH IMPROVED TIME RESOLUTION FOR SELECTED SEGMENTS VIA ADAPTIVE BLOCK TRANSFORMATION OF A GROUP OF SAMPLES FROM A SUBBAND DECOMPOSITION**

(75) Inventors: **Sanjeev Mehrotra**, Kirkland, WA (US); **Wei-Ge Chen**, Sammamish, WA (US); **Henrique Sarmento Malvar**, Sammamish, WA (US)

Correspondence Address:
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204 (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(21) Appl. No.: **11/183,271**

(22) Filed: **Jul. 15, 2005**

Publication Classification

(51) **Int. Cl.**
G10L 21/00 (2006.01)

(52) **U.S. Cl.** **704/203**

(57) **ABSTRACT**

A transform coder is described that performs a time-split transform in addition to a discrete cosine type transform. A time-split transform is selectively performed based on characteristics of media data. Transient detection identifies a changing signal characteristic, such as a transient in media data. After encoding an input signal from a time domain to a transform domain, a time-splitting transformer selectively perform an orthogonal sum-difference transform on adjacent coefficients indicated by a changing signal characteristic location. The orthogonal sum-difference transform on adjacent coefficients results in transforming a vector of coefficients in the transform domain as if they were multiplied by an identity matrix including at least one 2×2 time-split block along a diagonal of the matrix. A decoder performs an inverse of the described transforms.

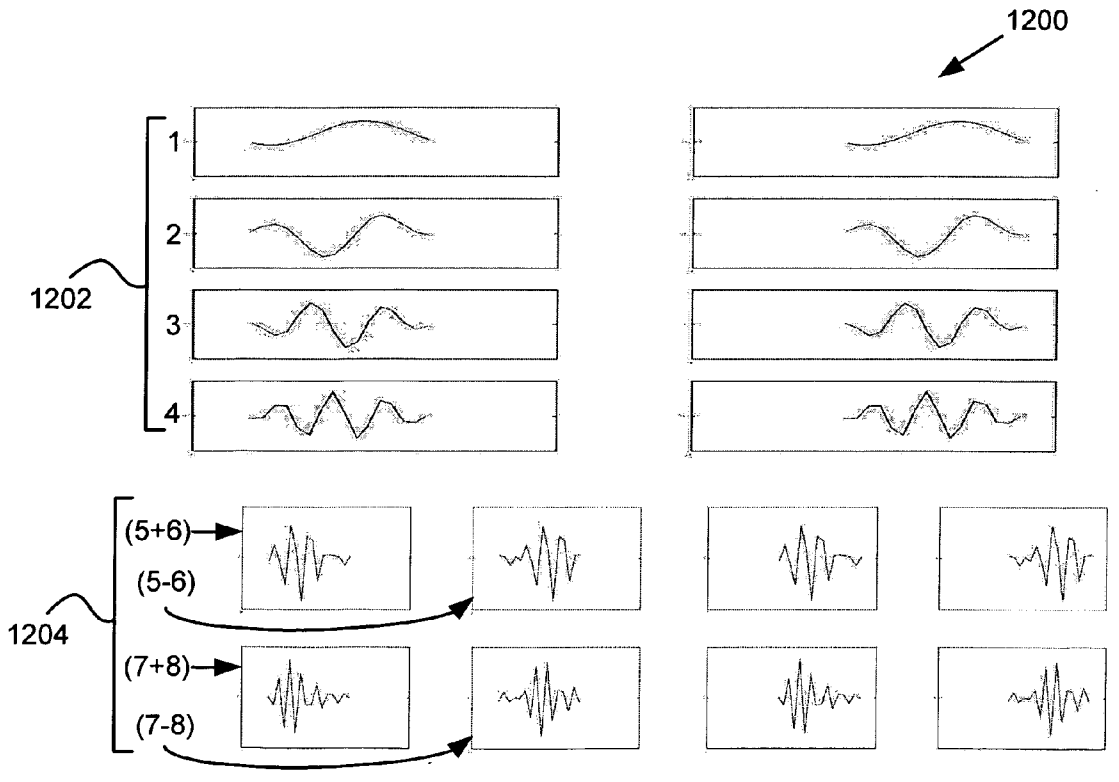


FIG. 1

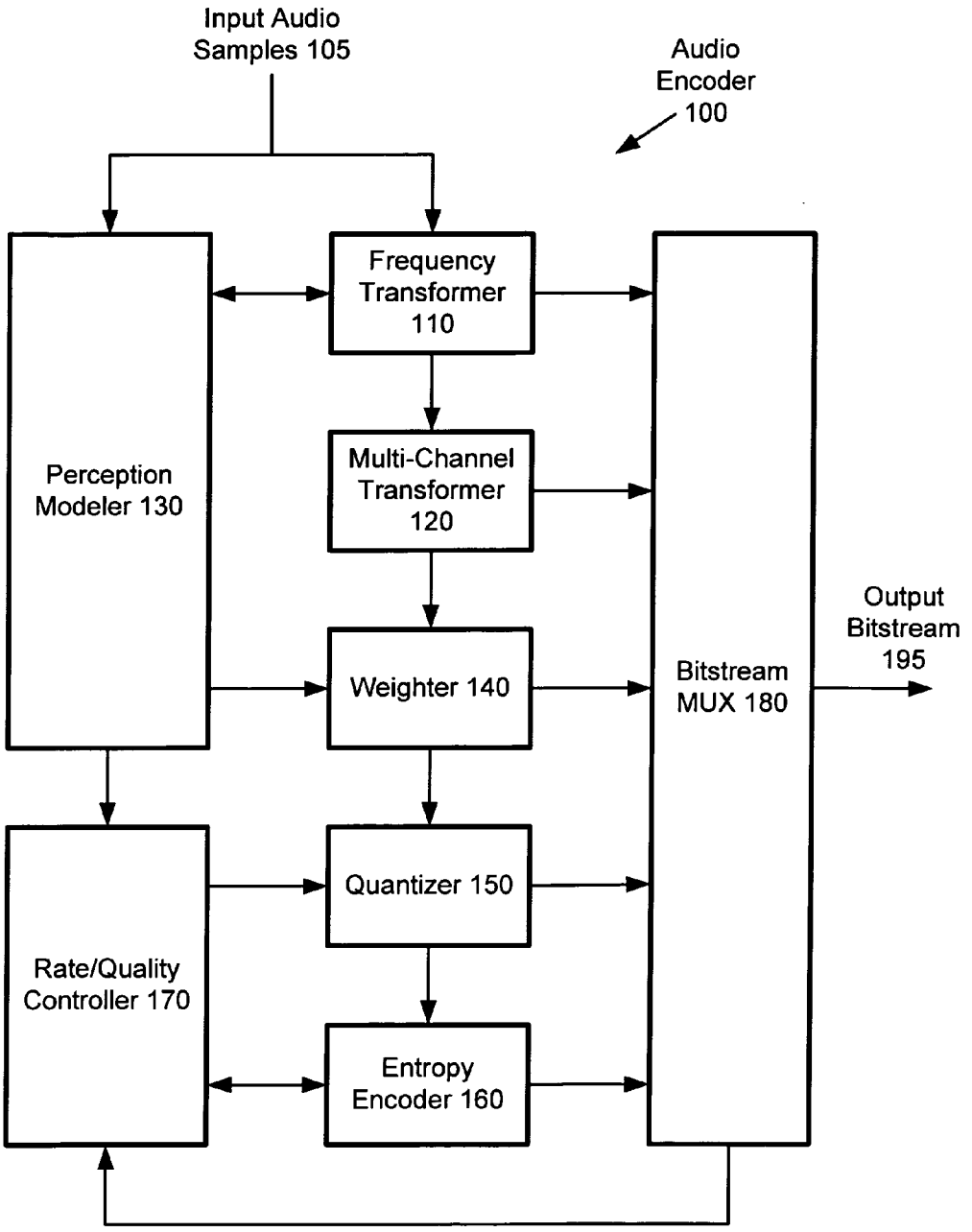

Input Audio Samples 105
Audio Encoder 100
Frequency Transformer 110
Multi-Channel Transformer 120
Perception Modeler 130
Weighter 140
Bitstream MUX 180
Output Bitstream 195
Quantizer 150
Rate/Quality Controller 170
Entropy Encoder 160

FIG. 2

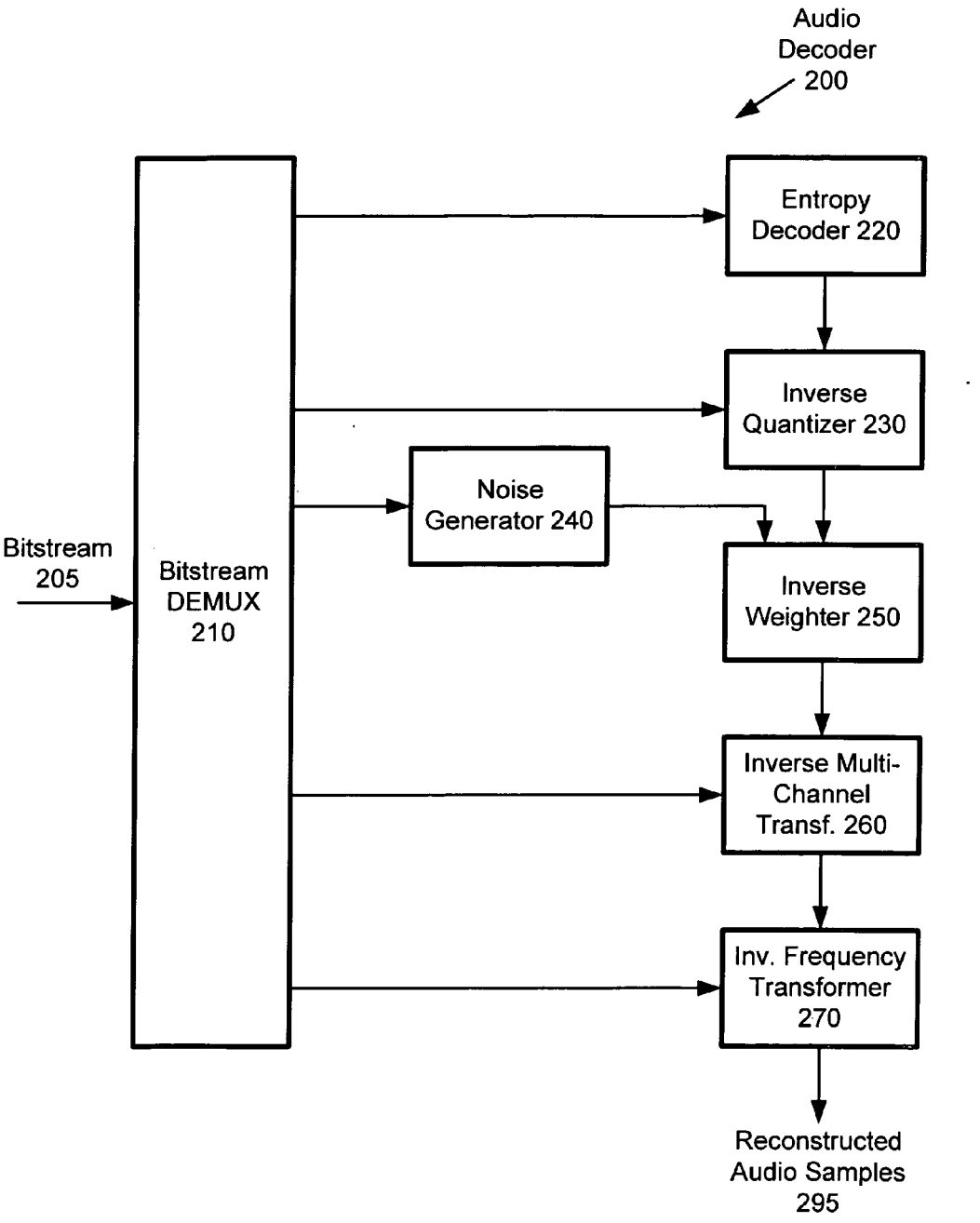

Audio Decoder 200
Bitstream 205
Bitstream DEMUX 210
Entropy Decoder 220
Inverse Quantizer 230
Noise Generator 240
Inverse Weighter 250
Inverse Multi-Channel Transf. 260
Inv. Frequency Transformer 270
Reconstructed Audio Samples 295

FIG. 3

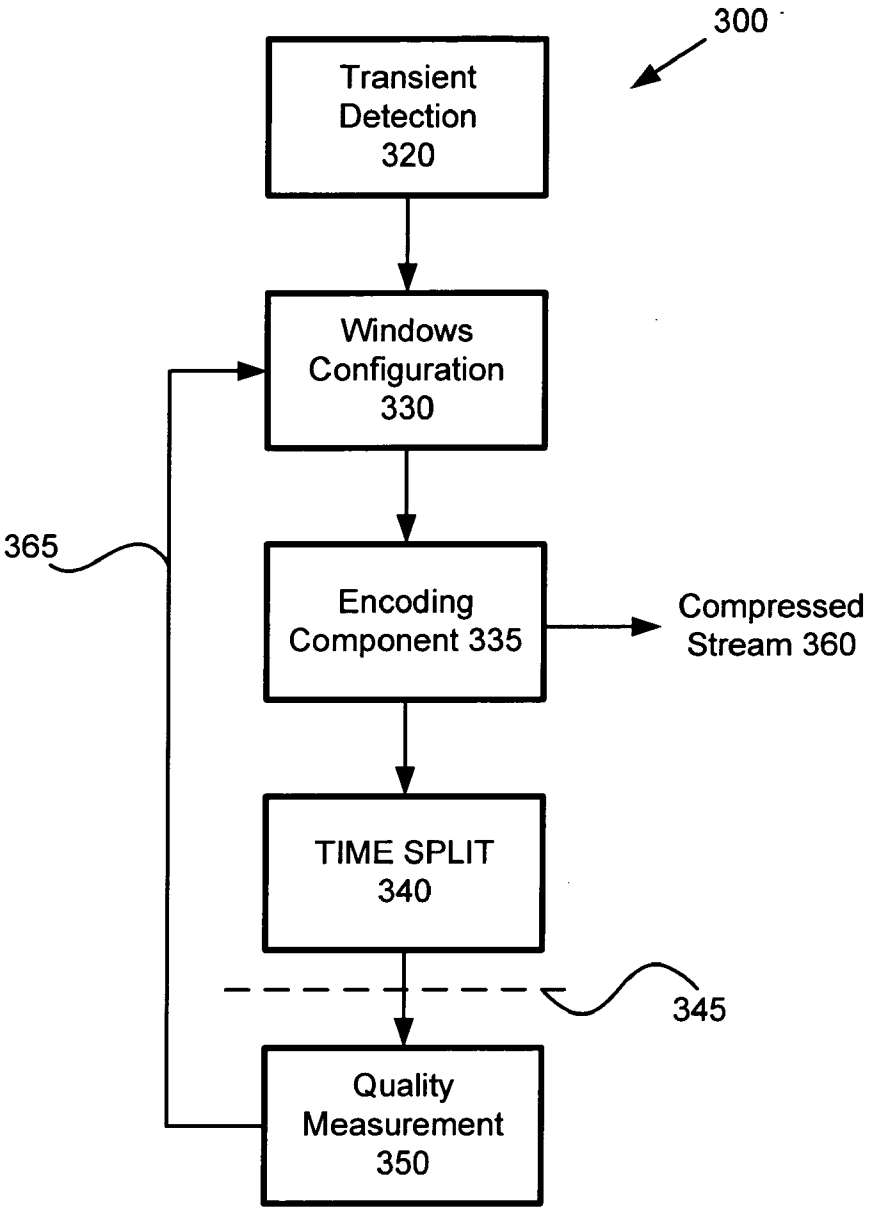

300
Transient Detection 320
Windows Configuration 330
365
Encoding Component 335
Compressed Stream 360
TIME SPLIT 340
345
Quality Measurement 350

FIG. 4

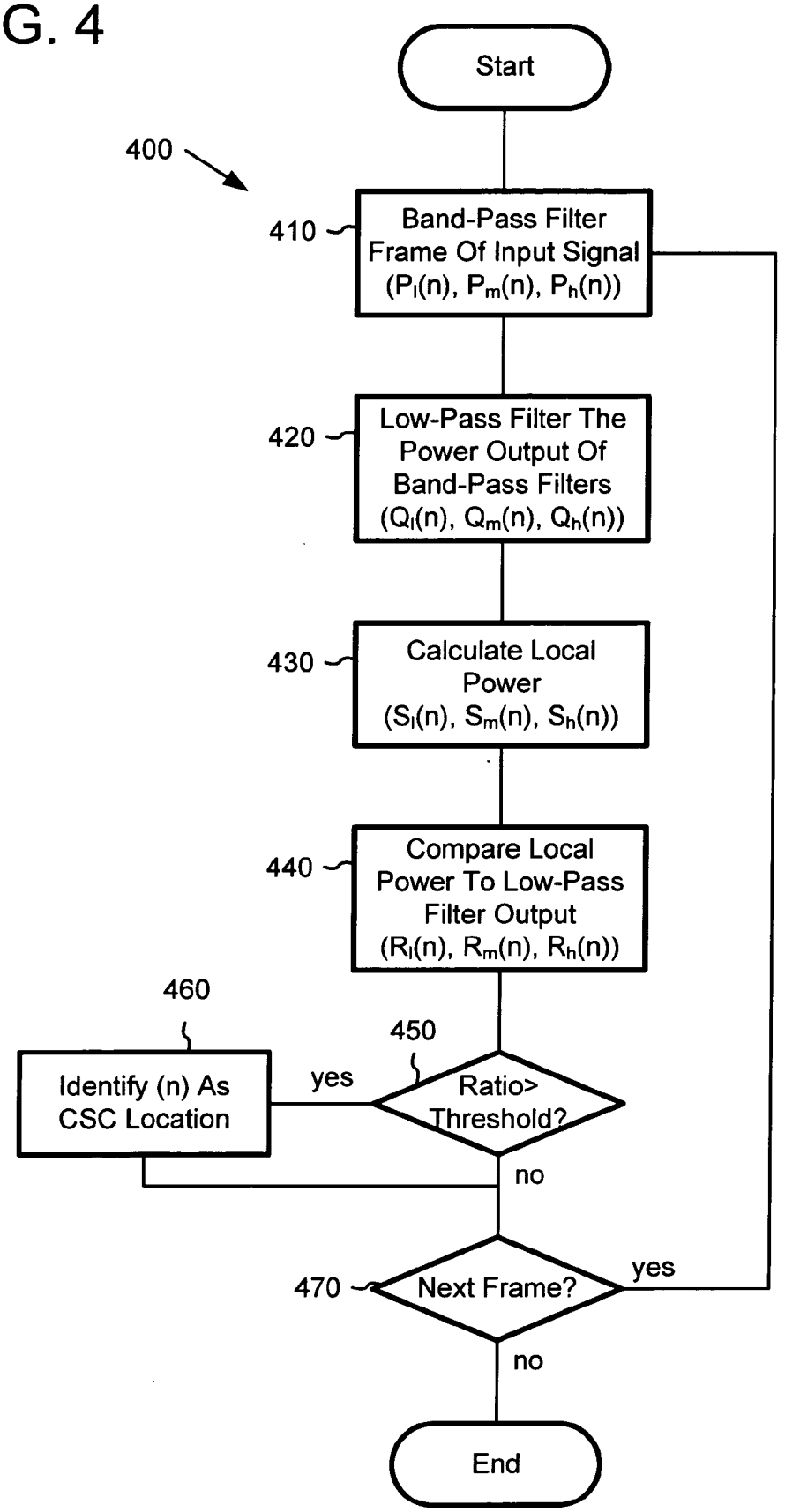

Start
400
410
Band-Pass Filter Frame Of Input Signal ($P_l(n)$, $P_m(n)$, $P_h(n)$)
420
Low-Pass Filter The Power Output Of Band-Pass Filters ($Q_l(n)$, $Q_m(n)$, $Q_h(n)$)
430
Calculate Local Power ($S_l(n)$, $S_m(n)$, $S_h(n)$)
440
Compare Local Power To Low-Pass Filter Output ($R_l(n)$, $R_m(n)$, $R_h(n)$)
460
450
Identify (n) As CSC Location
yes
Ratio> Threshold?
no
470
Next Frame?
yes
no
End

FIG. 5

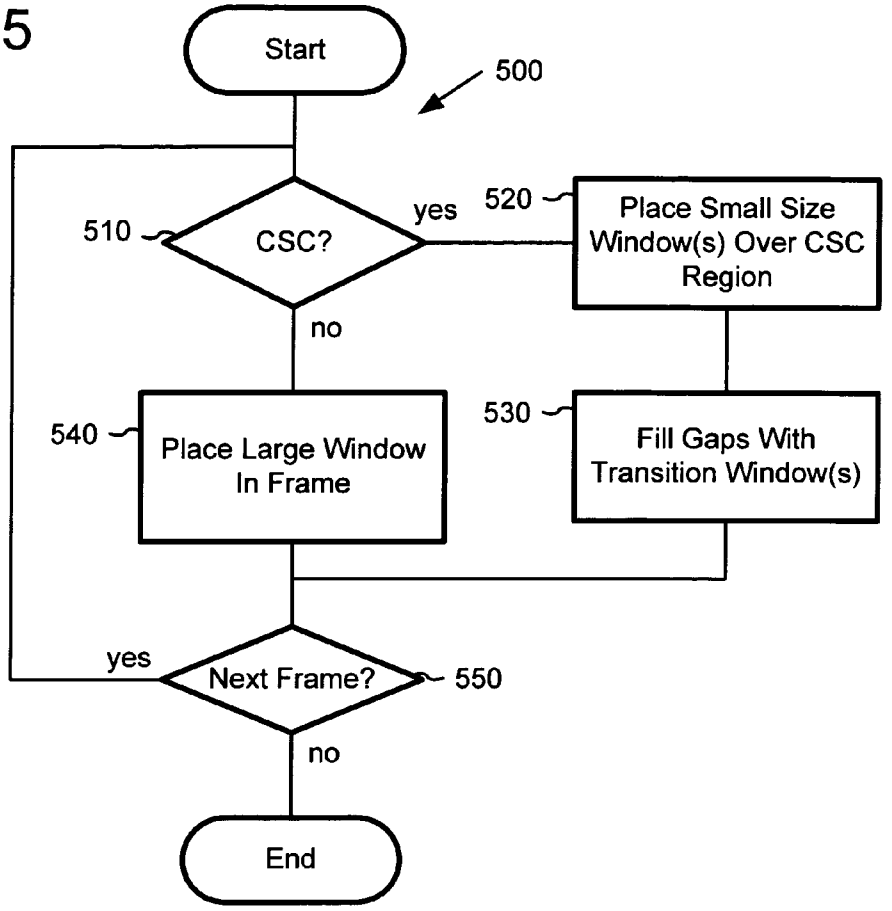

Start
500
510
CSC?
yes
520
Place Small Size Window(s) Over CSC Region
no
540
Place Large Window In Frame
530
Fill Gaps With Transition Window(s)
yes
Next Frame?
550
no
End

FIG. 6

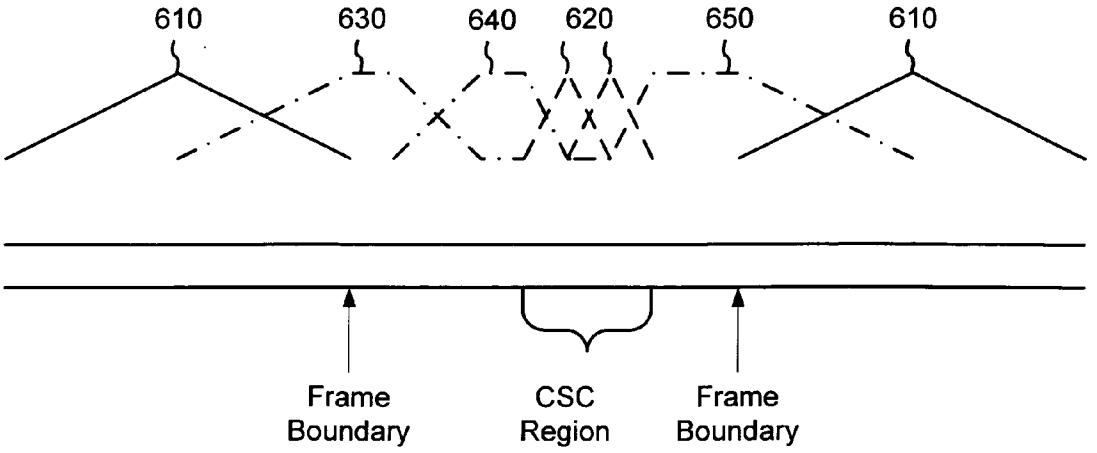

610
630
640
620
650
610
Frame Boundary
CSC Region
Frame Boundary

FIG. 7

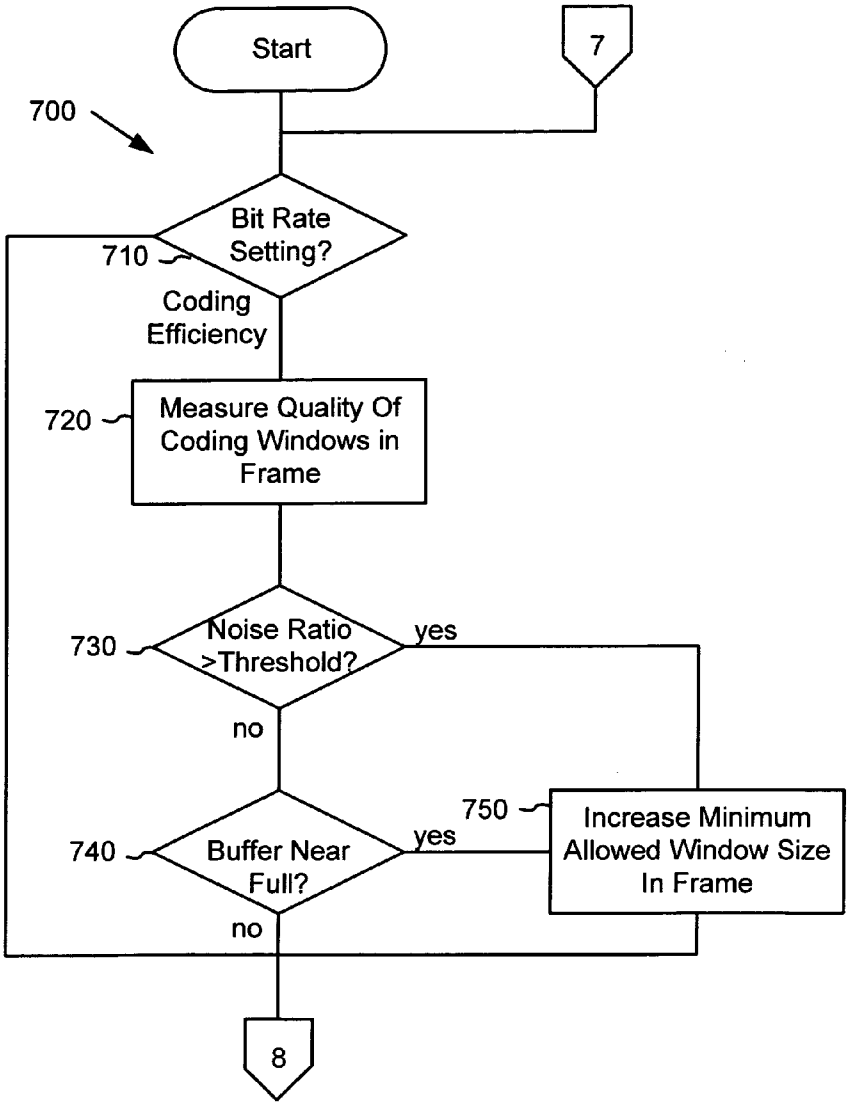

Start
7
700
Bit Rate Setting?
710
Coding Efficiency
720
Measure Quality Of Coding Windows in Frame
730
Noise Ratio >Threshold?
yes
no
740
Buffer Near Full?
yes
750
Increase Minimum Allowed Window Size In Frame
no
8

FIG. 8

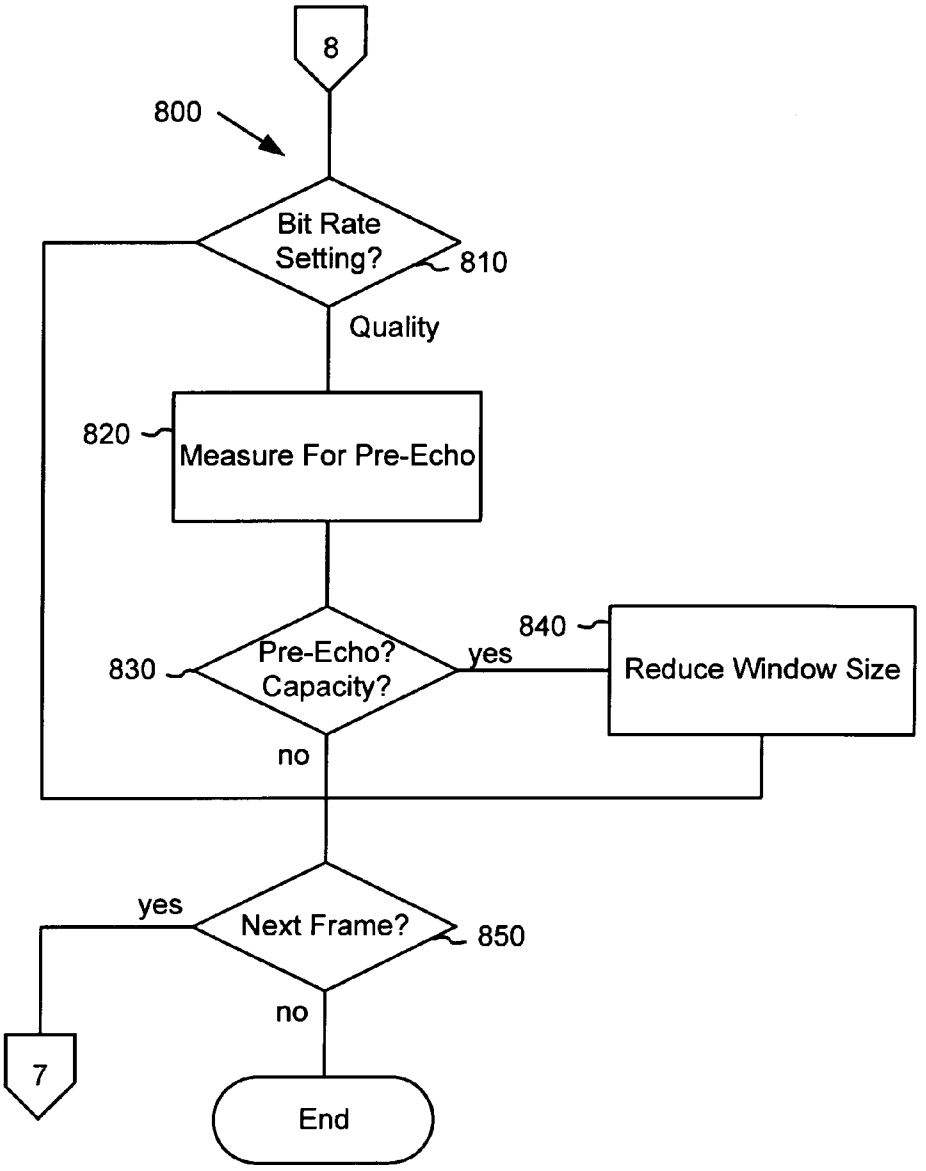

8
800
Bit Rate Setting?
810
Quality
820
Measure For Pre-Echo
830
Pre-Echo? Capacity?
yes
840
Reduce Window Size
no
yes
Next Frame?
850
no
7
End

FIG. 9

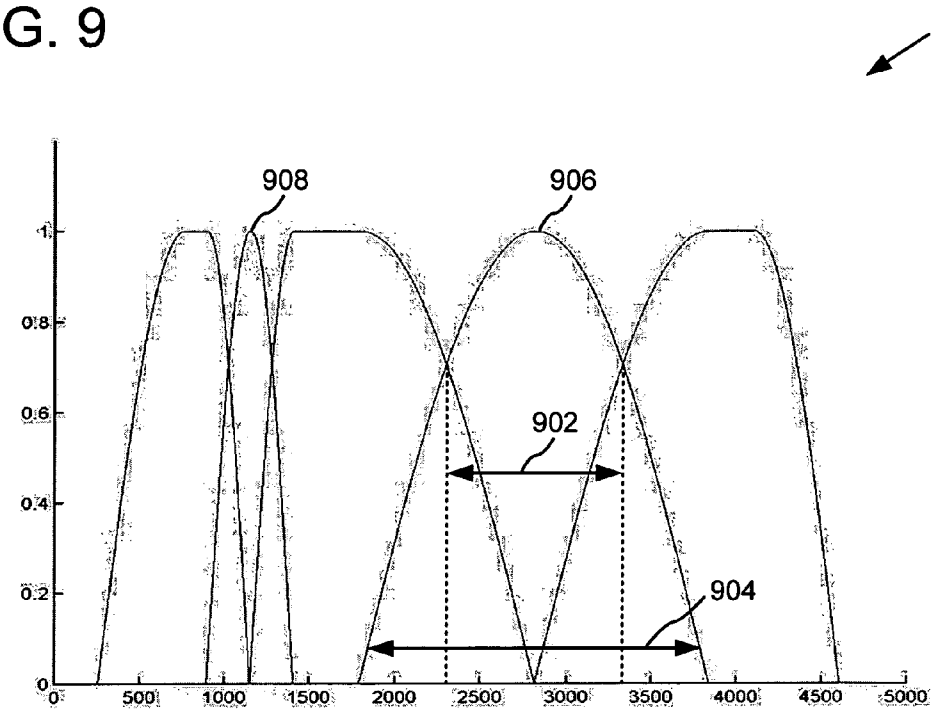

900
908
906
902
904
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

FIG. 10

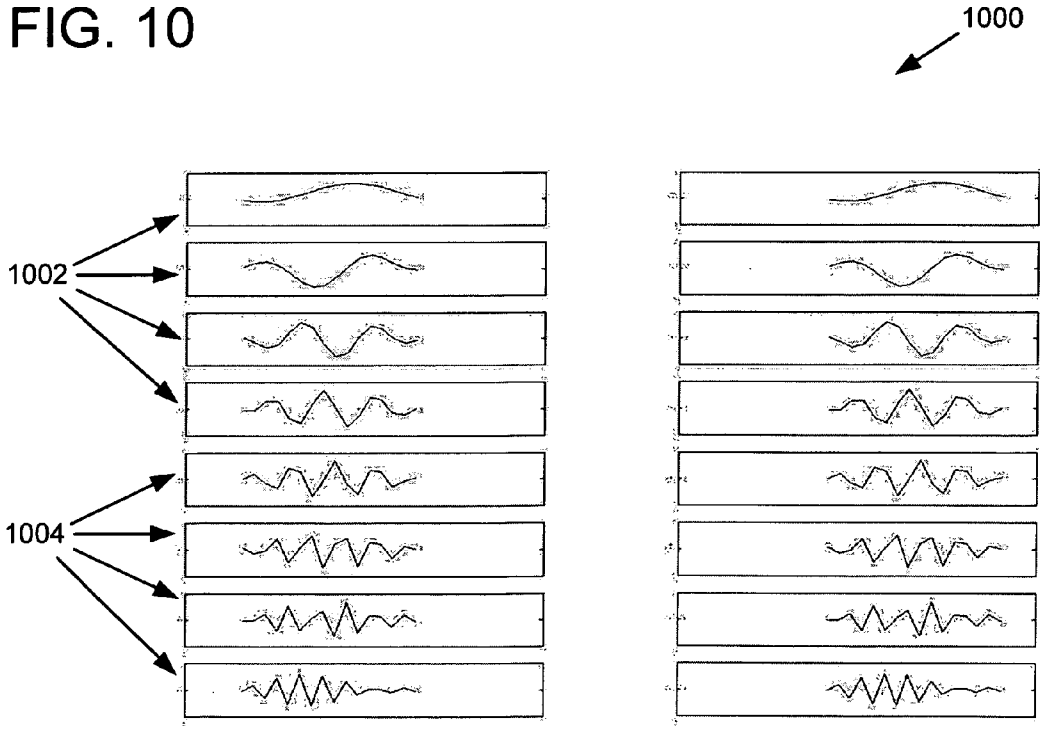

1000
1002
1004

FIG. 11

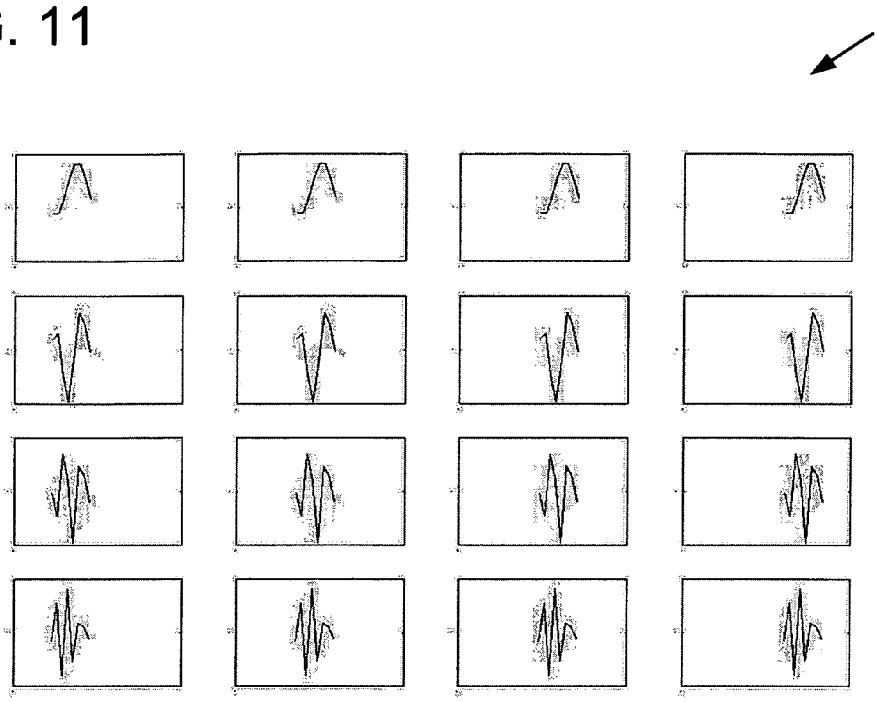


FIG. 12

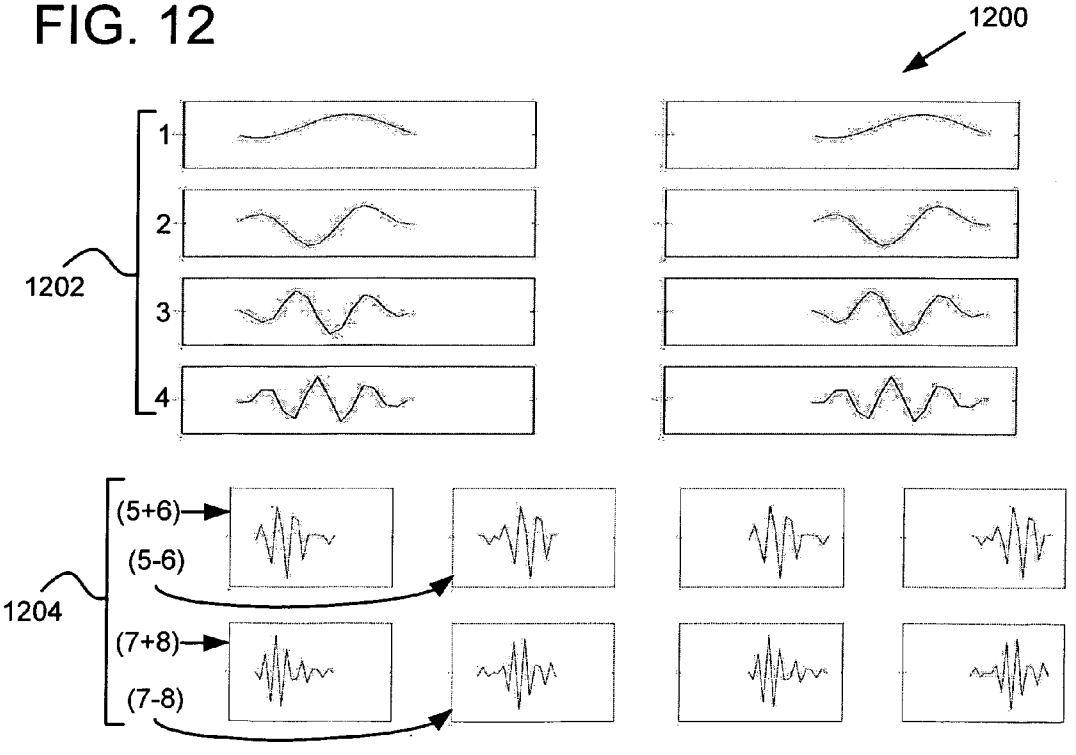
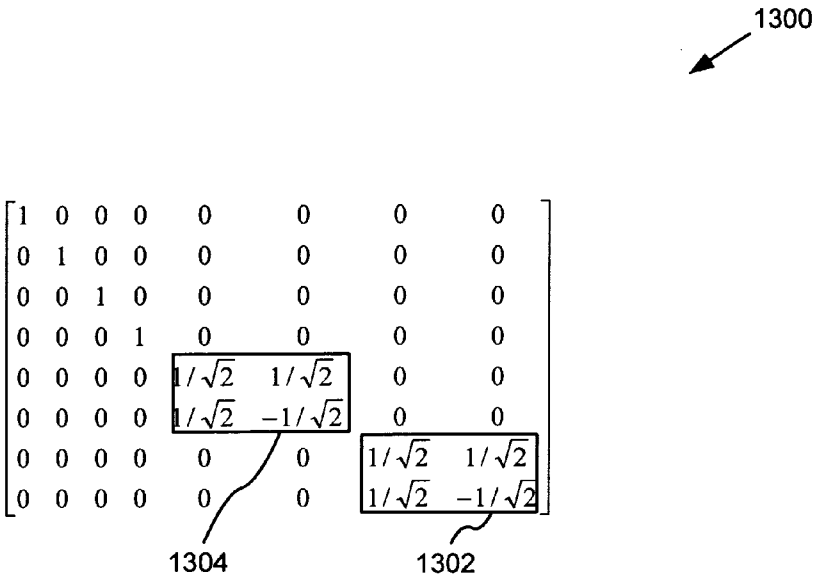


FIG. 13



$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

FIG. 14

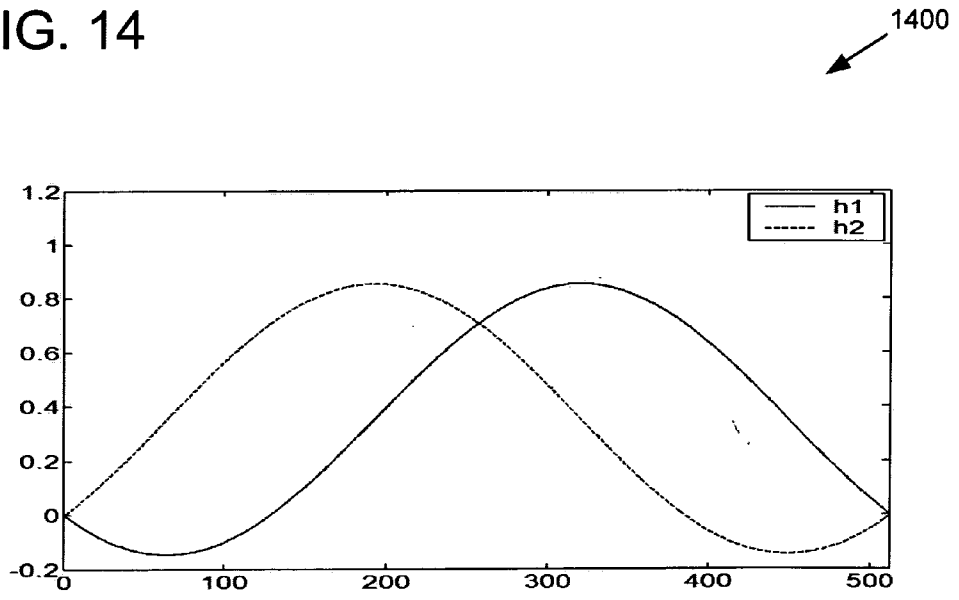


FIG. 15

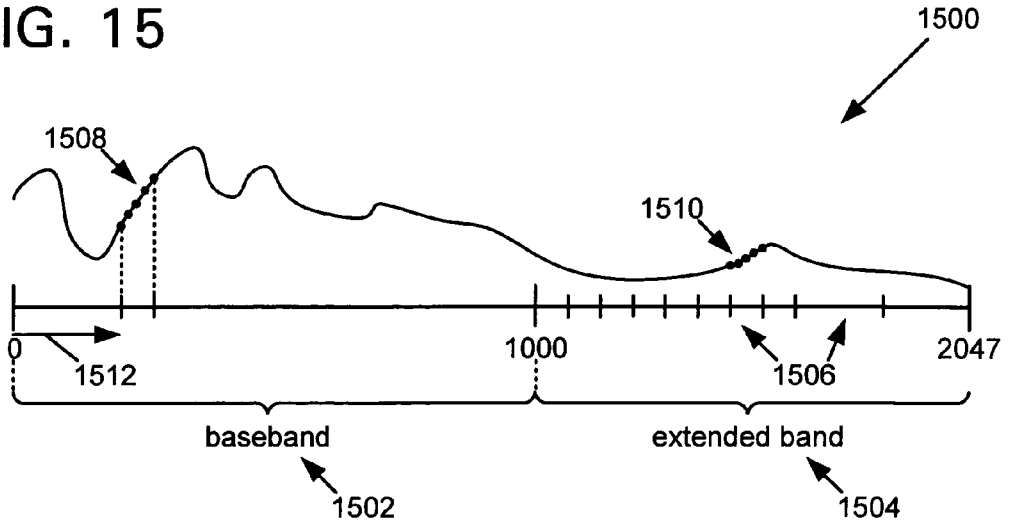

1500
1508
1510
0
1512
1000
1506
2047
baseband
extended band
1502
1504

FIG. 16

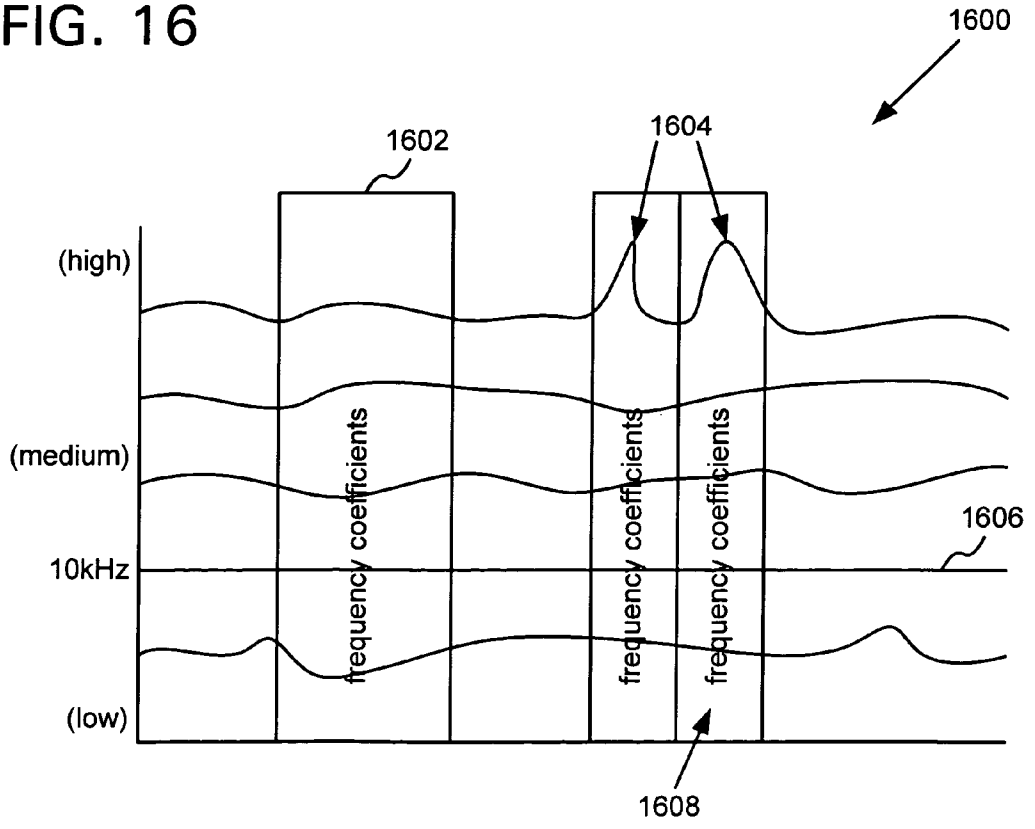

1600
1602
1604
(high)
(medium)
10kHz
(low)
frequency coefficients
frequency coefficients
frequency coefficients
1606
1608

FIG. 17

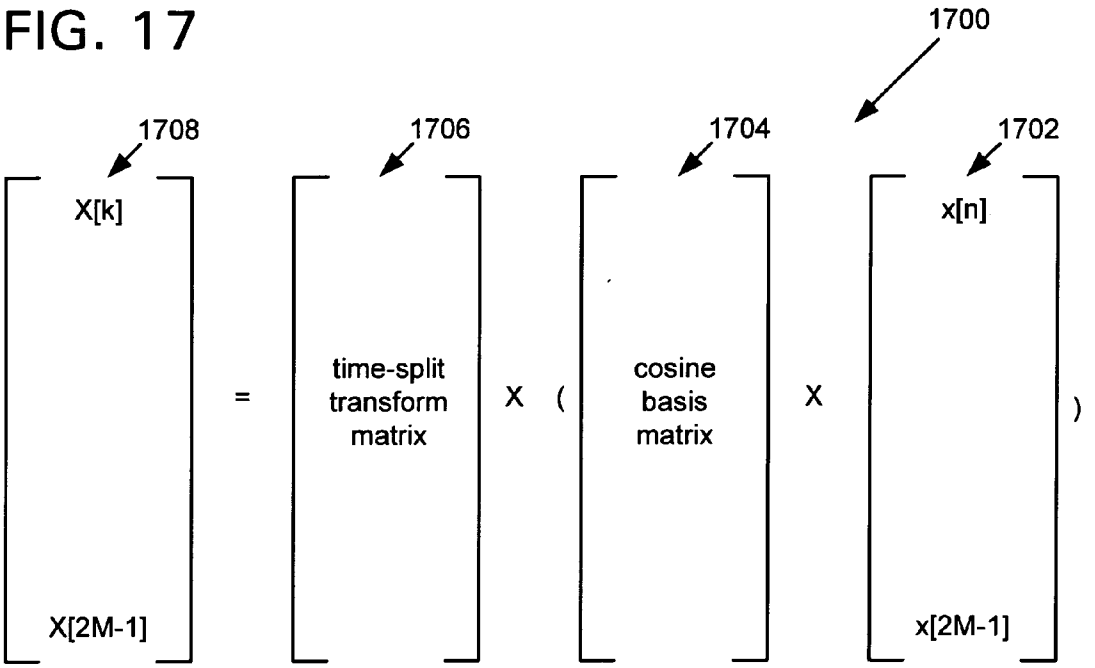
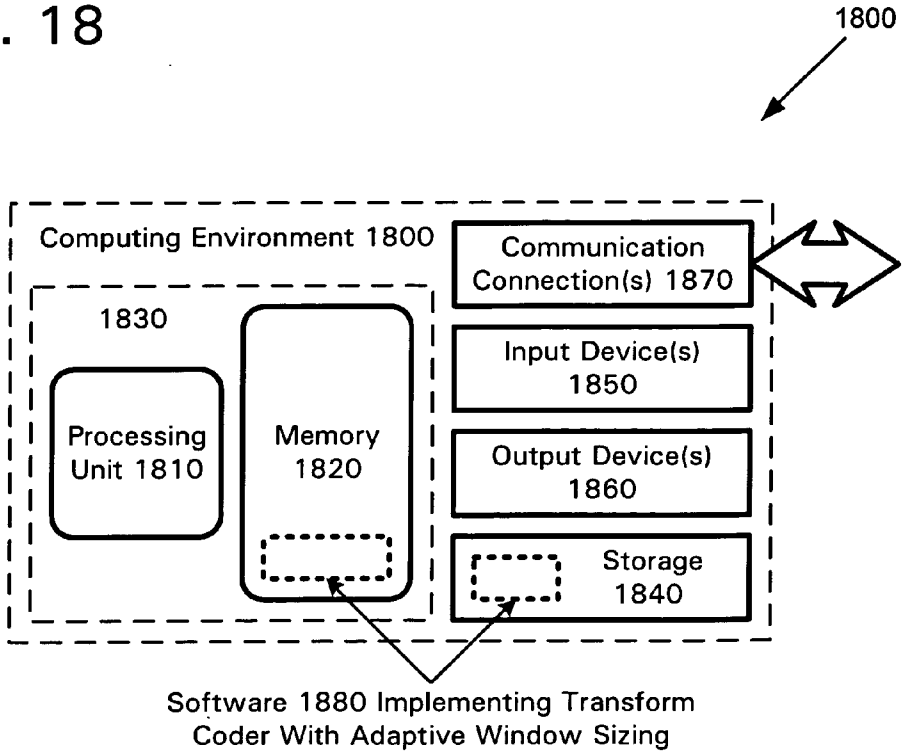

1700
1708
1706
1704
1702
X[k]
X[2M-1]
=
time-split transform matrix
X
(
cosine basis matrix
X
x[n]
x[2M-1]
)

FIG. 18


1800
Computing Environment 1800
1830
Processing Unit 1810
Memory 1820
Communication Connection(s) 1870
Input Device(s) 1850
Output Device(s) 1860
Storage 1840
Software 1880 Implementing Transform Coder With Adaptive Window Sizing

CODING WITH IMPROVED TIME RESOLUTION FOR SELECTED SEGMENTS VIA ADAPTIVE BLOCK TRANSFORMATION OF A GROUP OF SAMPLES FROM A SUBBAND DECOMPOSITION

BACKGROUND

[0001] Transform coding is a compression technique often used in digital media compression systems. Uncompressed digital media, such as an audio or video signal is typically represented as a stream of amplitude samples of a signal taken at regular time intervals. For example, a typical format for audio on compact disks consists of a stream of sixteen-bit samples per channel of the audio (e.g., the original analog audio signal from a microphone) captured at a rate of 44.1 KHz. Each sample is a sixteen-bit number representing the amplitude of the audio signal at the time of capture. Other digital media systems may use various different amplitude and time resolutions of signal sampling.

[0002] Uncompressed digital media can consume considerable storage and transmission capacity. Transform coding reduces the size of digital media by transforming the time-domain representation of the digital media into a frequency-domain (or other like transform domain) representation, and then reducing resolution of certain generally less perceptible frequency components of the frequency-domain representation. This generally produces much less perceptible degradation of the signal compared to reducing amplitude or time resolution of digital media in the time domain.

[0003] More specifically, a typical audio transform coding technique divides the uncompressed digital audio's stream of time-samples into fixed-size subsets or blocks, each block possibly overlapping with other blocks. A linear transform that does time-frequency analysis is applied to each block, which converts the time interval audio samples within the block to a set of frequency (or transform) coefficients generally representing the strength of the audio signal in corresponding frequency bands over the block interval. For compression, the transform coefficients may be selectively quantized (i.e., reduced in resolution, such as by dropping least significant bits of the coefficient values or otherwise mapping values in a higher resolution number set to a lower resolution), and also entropy or variable-length coded into a compressed audio data stream. At decoding, the transform coefficients will inversely transform to nearly reconstruct the original amplitude/time sampled audio signal.

[0004] Many audio compression systems utilize the Modulated Lapped Transform (MLT, also known as Modified Discrete Cosine Transform or MDCT) to perform the time-frequency analysis in audio transform coding. MLT reduces blocking artifacts introduced into the reconstructed audio signal by quantization. More particularly, when non-overlapping blocks are independently transform coded, quantization errors will produce discontinuities in the signal at the block boundaries upon reconstruction of the audio signal at the decoder.

[0005] One problem in audio coding is commonly referred to as "pre-echo." Pre-echo occurs when the audio undergoes a sudden change (referred to as a "changing signal characteristic"). For example, a changing signal characteristic such as a transient. In transform coding, particular frequency coefficients commonly are quantized (i.e., reduced in resolution). When the transform coefficients are later inverse-transformed to reproduce the audio signal, this quantization introduces quantization noise that is spread over the entire block in the time domain. This inherently causes rather uniform smearing of noise within the coding frame. The noise, which generally is tolerable for some part of the frame, can be audible and disastrous to auditory quality during portions of the frame where the masking level is low. In practice, this effect shows up most prominently when a signal has a sharp attack immediately following a region of low energy, hence the term "pre-echo."“Post-echo" is a changing signal characteristic that occurs when the signal transition from high to low energy is less of a problem to perceptible auditory quality due to a property of the human auditory system.

[0006] Thus, what is needed is a system that addresses the pre-echo effect by reducing the smearing of quantization noise over a large signal frame.

SUMMARY

[0007] A transform coder is described that performs an additional time-split transform selectively based on characteristics of media data. A transient detection component identifies changing signal characteristic locations, such as transient locations to apply a time-split transform. For example, a slow transition between two types of signals is usually not considered a transient and yet the described technology provides benefits for such changing signal characteristics. An encoding component transforms an input signal from a time domain to a transform domain. A time-splitting transformer component selectively performs an orthogonal sum-difference transform on adjacent coefficients indicated by the identified changing signal characteristic location. The orthogonal sum/difference transform results in transforming a vector of coefficients in the transform domain as if they were multiplied selectively by one or more exemplary time-split transform matrices.

[0008] In other examples, a window configuration component configures window sizes so as to place one or more small window sizes in areas of transient locations and large window sizes in other areas. The encoding component inverse-transforms to produce a reconstructed version of the input signal and a quality measurement component measures the achieved quality of the reconstructed signal. The window configuration component adjusts window sizes according to the achieved quality. The quality measurement component further operates to measure achieved perceptual quantization noise of the reconstructed signal. The window configuration component further operates to increase a window size where the measure of achieved perceptual quantization noise exceeds an acceptable threshold. The quality measurement component further operates to detect pre-echo in the reconstructed signal and the window configuration component further operates to decrease window size where pre-echo is detected.

[0009] A transform decoder provides an inverse time-splitting transformer and an inverse transformer. The inverse time-splitting transformer receives side information and coefficient data in a transform domain and selectively performs an inverse orthogonal sum-difference transformation on adjacent coefficients indicated in received side information. Next, the inverse transformer transforms coefficient data from the transform domain to a time domain.

[0010] In other examples, an inverse window configuration component receives side information about window and sub-frame sizes and the inverse transformer transforms coefficient data according to the window and sub-band sizes. In one such example, the inverse orthogonal sum-difference transformation results in transforming a vector of coefficients in the transform domain as if it were multiplied by an inverse of a time-splitting transform. In another example, the inverse time-splitting transformer component receives side information indicating that there are no time-splits in at least one sub-frame, and in another example, the side information indicates whether or not there is a time-split in an extended band.

[0011] A method of decoding receives side information and coefficient data in a transform domain. The method selectively performs an inverse time-split transform on adjacent coefficients as indicated in received side information and further transforms the coefficient data from the transform domain to a time domain. In another example, the method identifies sub-frame sizes in received side information and the inverse transform is performed according to the identified sub-frame sizes. In yet another example, the side information indicates whether there is a time-split in a sub-band, or whether or not there is a time-split in each sub-band in an extended band. In another example, the method determines a pair of adjacent coefficients in a transform domain on which to perform an inverse sum-difference transform.

[0012] Additional features and advantages of the invention will be made apparent from the following detailed description of embodiments that proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. **1** is a block diagram of an exemplary audio encoder performing selective time-split transform.

[0014] FIG. **2** is a block diagram of an exemplary audio decoder performing inverse selective time-split transform.

[0015] FIG. **3** is a block diagram of an exemplary transform coder performing selective time-split transform.

[0016] FIG. **4** is a flow chart of an exemplary changing signal characteristic detection process.

[0017] FIG. **5** is a flow chart of an exemplary window configuration process.

[0018] FIG. **6** is a graph of an example window configuration produced via the process of FIG. **5**.

[0019] FIG. **7** is a flow chart of an exemplary windows configuration process.

[0020] FIG. **8** is a flow chart of an exemplary process to detect pre-echo.

[0021] FIG. **9** is a graph representing exemplary overlapping windows covering segmentation blocks.

[0022] FIG. **10** is a graph of the basis vectors that contribute to the MLT coefficients corresponding to the middle two sub-frames.

[0023] FIG. **11** is a graph of the basis vectors that contribute to the MLT coefficients corresponding to the middle four sub-frames with smaller sized segmentation.

[0024] FIG. **12** is a graph representing how time-splitting combines adjacent coefficients.

[0025] FIG. **13** is a matrix representing an exemplary time-split transform of FIG. **12**.

[0026] FIG. **14** is a graph of two new exemplary time-split window functions.

[0027] FIG. **15** is a graph representing an exemplary set of spectral coefficients.

[0028] FIG. **16** is a graph of an exemplary time-frequency plot of selected frequency coefficients.

[0029] FIG. **17** is a diagram representing a linear transformation of a time domain vector into a transform domain vector including a time-split transform matrix of FIG. **13**.

[0030] FIG. **18** illustrates a generalized example of a suitable computing environment in which the illustrative embodiment may be implemented.

DETAILED DESCRIPTION

Brief Overview

[0031] The following describes a transform coder capable of performing an additional time-split transform selectively based on characteristics of spectral digital media data.

[0032] Optionally, an adaptive window size is provided when a selective time-split transform does not produce a sufficient benefit. The coder selects one or more window sizes within a frame of spectral digital media data. Spectral Data analysis (e.g., changing signal characteristic detection) identifies one or more frequencies for a time-split transform. If the results of a time-split transform are not sufficient, then a window size may be adapted. Optionally, using one or more passes at time-split transform, data energy analysis, and or window size adaptation provides improved coding efficiency overall.

[0033] When providing a sub-band decomposition for coding of data, with overlapped or block based transform, or when using a filterbank (which can also be represented as an overlapped transform), the sub-band structure is typically fixed. When providing an overlapped transform (such as modulated lapped transform (MLT)), the sub-frame size can be varied which results in adapting the time/frequency resolution depending on signal characteristics. However, there are certain cases in which using a large sub-frame size (better frequency resolution, lower time resolution) provides efficient coding, but results in noticeable artifacts at higher frequencies. In order to remove these artifacts, various possible features are described for reducing artifacts. For example, a block based transform (e.g., a time-split transform) is applied subsequent to an existing fixed transform (e.g., a discrete cosine (DCT transform, a MLT transform, etc.)). In one example, the time-split transform is used selectively to provide better time resolution upon determining that the time-split transform is beneficial for one or more select groups of frequency coefficients. The frequency selections is based on detected energy change.

[0034] If there are only certain regions of the frequency that need better time resolution, then using a smaller time window can result in a significant increase in the number of bits needed to code the spectral data. If sufficient bits are

available this is not an issue, and a smaller time window should be used. However, when there are not enough bits, using a selective time-split transform on only those frequency ranges where it is needed can provide improved quality.

[0035] A time-split transform improves data coding when better time resolution is needed for coding of certain frequencies. A time-split transform and or various other features described herein can be used in any media encoder or decoder. For example, a time-split transform can be used with the digital media codec techniques described by Mehrotra et. al., "Efficient Coding of Digital Media Spectral Data Using Wide-Sense Perceptual Similarity" U.S. patent application Ser. No. 10/882,801, filed Jun. 29, 2004. For example, a time-split transform can be used to improve coding of high, medium, or low frequencies.

Exemplary Encoder and Decoder

[0036] FIG. **1** is a block diagram of a generalized audio encoder (**100**). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, divided into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoders or decoders with different modules and/or other configurations of modules perform time-split transforms.

[0037] The generalized audio encoder (**100**) includes a frequency transformer (**110**), a multi-channel transformer (**120**), a perception modeler (**130**), a weighter (**140**), a quantizer (**150**), an entropy encoder (**160**), a rate/quality controller (**170**), and a bitstream multiplexer ["MUX"] (**180**).

[0038] The encoder (**100**) receives a time series of input audio samples (**105**). For input with multiple channels (e.g., stereo mode), the encoder (**100**) processes channels independently, and can work with jointly coded channels following the multi-channel transformer (**120**). The encoder (**100**) compresses the audio samples (**105**) and multiplexes information produced by the various modules of the encoder (**100**) to output a bitstream (**195**) in a format such as Windows Media Audio ["WMA"] or Advanced Streaming Format ["ASF"]. Alternatively, the encoder (**100**) works with other input and/or output formats.

[0039] The frequency transformer (**110**) receives the audio samples (**105**) and converts them into data in the frequency domain. The frequency transformer (**110**) splits the audio samples (**105**) into blocks, which can have variable size to allow variable temporal resolution. Small blocks allow for greater preservation of time detail at short but active transition segments in the input audio samples (**105**), but sacrifice some frequency resolution. In contrast, large blocks have better frequency resolution and worse time resolution, and usually allow for greater compression efficiency at longer and less active segments. Blocks can overlap to reduce perceptible discontinuities between blocks that could otherwise be introduced by later quantization. The frequency transformer selectively applies a time-split transform based on characteristics of the data. The frequency transformer (**110**) outputs blocks of frequency coefficient data to the multi-channel transformer (**120**) and outputs side information such as block sizes to the MUX (**180**). The frequency transformer (**110**) outputs both the frequency coefficient data and the side information to the perception modeler (**130**).

[0040] The frequency transformer (**110**) partitions a frame of audio input samples (**105**) into overlapping sub-frame blocks with time-varying size and applies a time-varying MLT to the sub-frame blocks. Possible sub-frame sizes include 128, 256, 512, 1024, 2048, and 4096 samples. The MLT operates like a DCT modulated by a time window function, where the window function is time varying and depends on the sequence of sub-frame sizes. The MLT transforms a given overlapping block of samples x[n], 0≦n<subframe_size into a block of frequency coefficients X[k],0≦k<subframe_size/2 The frequency transformer (**110**) can also output estimates of the complexity of future frames to the rate/quality controller (**170**). Alternative embodiments use other varieties of MLT. In still other alternative embodiments, the frequency transformer (**110**) applies a DCT, FFT, or other type of modulated or non-modulated, overlapped or non-overlapped frequency transform, or use subband or wavelet coding. Typically after the transform to the frequency domain, the frequency transformer selectively applies a time-split transform based on characteristics of the data.

[0041] For multi-channel audio data, the multiple channels of frequency coefficient data produced by the frequency transformer (**110**) often correlate. To exploit this correlation, the multi-channel transformer (**120**) can convert the multiple original, independently coded channels into jointly coded channels. For example, if the input is stereo mode, the multi-channel transformer (**120**) can convert the left and right channels into sum and difference channels:

$$X_{Sum}[k] = \frac{X_{Left}[k] + X_{Right}[k]}{2}$$

$$X_{Diff}[k] = \frac{X_{Left}[k] - X_{Right}[k]}{2}$$

[0042] Or, the multi-channel transformer (**120**) can pass the left and right channels through as independently coded channels. More generally, for a number of input channels greater than one, the multi-channel transformer (**120**) passes original, independently coded channels through unchanged or converts the original channels into jointly coded channels. The decision to use independently or jointly coded channels can be predetermined, or the decision can be made adaptively on a block by block or other basis during encoding. The multi-channel transformer (**120**) produces side information to the MUX (**180**) indicating the channel mode used.

[0043] The perception modeler (**130**) models properties of the human auditory system to improve the quality of the reconstructed audio signal for a given bitrate. The perception modeler (**130**) computes the excitation pattern of a variable-size block of frequency coefficients. First, the perception modeler (**130**) normalizes the size and amplitude scale of the block. This enables subsequent temporal smearing and establishes a consistent scale for quality measures. Optionally, the perception modeler (**130**) attenuates the coefficients

at certain frequencies to model the outer/middle ear transfer function. The perception modeler (**130**) computes the energy of the coefficients in the block and aggregates the energies by 25 critical bands. Alternatively, the perception modeler (**130**) uses another number of critical bands (e.g., 55 or 109). The frequency ranges for the critical bands are implementation-dependent, and numerous options are well known. For example, see ITU-R BS **1387** or a reference mentioned therein. The perception modeler (**130**) processes the band energies to account for simultaneous and temporal masking. In alternative embodiments, the perception modeler (**130**) processes the audio data according to a different auditory model, such as one described or mentioned in ITU-R BS **1387**.

[**0044**] The weighter (**140**) generates weighting factors (alternatively called a quantization matrix) based upon the excitation pattern received from the perception modeler (**130**) and applies the weighting factors to the data received from the multi-channel transformer (**120**). The weighting factors include a weight for each of multiple quantization bands in the audio data. The quantization bands can be the same or different in number or position from the critical bands used elsewhere in the encoder (**100**). The weighting factors indicate proportions at which noise is spread across the quantization bands, with the goal of minimizing the audibility of the noise by putting more noise in bands where it is less audible, and vice versa. The weighting factors can vary in amplitudes and number of quantization bands from block to block. In one implementation, the number of quantization bands varies according to block size; smaller blocks have fewer quantization bands than larger blocks. For example, blocks with 128 coefficients have 13 quantization bands, blocks with 256 coefficients have 15 quantization bands, up to 25 quantization bands for blocks with 2048 coefficients. The weighter (**140**) generates a set of weighting factors for each channel of multi-channel audio data in independently coded channels, or generates a single set of weighting factors for jointly coded channels. In alternative embodiments, the weighter (**140**) generates the weighting factors from information other than or in addition to excitation patterns.

[**0045**] The weighter (**140**) outputs weighted blocks of coefficient data to the quantizer (**150**) and outputs side information such as the set of weighting factors to the MUX (**180**). The weighter (**140**) can also output the weighting factors to the rate/quality controller (**140**) or other modules in the encoder (**100**). The set of weighting factors can be compressed for more efficient representation. If the weighting factors are lossy compressed, the reconstructed weighting factors are typically used to weight the blocks of coefficient data. If audio information in a band of a block is completely eliminated for some reason (e.g., noise substitution or band truncation), the encoder (**100**) may be able to further improve the compression of the quantization matrix for the block.

[**0046**] The quantizer (**150**) quantizes the output of the weighter (**140**), producing quantized coefficient data to the entropy encoder (**160**) and side information including quantization step size to the MUX (**180**). Quantization introduces irreversible loss of information, but also allows the encoder (**100**) to regulate the bitrate of the output bitstream (**195**) in conjunction with the rate/quality controller (**170**). In FIG. **1**, the quantizer (**150**) is an adaptive, uniform scalar quantizer. The quantizer (**150**) applies the same quantization step size to each frequency coefficient, but the quantization step size itself can change from one iteration to the next to affect the bitrate of the entropy encoder (**160**) output. In alternative embodiments, the quantizer is a non-uniform quantizer, a vector quantizer, and/or a non-adaptive quantizer.

[**0047**] The entropy encoder (**160**) losslessly compresses quantized coefficient data received from the quantizer (**150**). For example, the entropy encoder (**160**) uses multi-level run length coding, variable-to-variable length coding, run length coding, Huffman coding, dictionary coding, arithmetic coding, LZ coding, a combination of the above, or some other entropy encoding technique.

[**0048**] The rate/quality controller (**170**) works with the quantizer (**150**) to regulate the bitrate and quality of the output of the encoder (**100**). The rate/quality controller (**170**) receives information from other modules of the encoder (**100**). In one implementation, the rate/quality controller (**170**) receives estimates of future complexity from the frequency transformer (**110**), sampling rate, block size information, the excitation pattern of original audio data from the perception modeler (**130**), weighting factors from the weighter (**140**), a block of quantized audio information in some form (e.g., quantized, reconstructed, or encoded), and buffer status information from the MUX (**180**). The rate/quality controller (**170**) can include an inverse quantizer, an inverse weighter, an inverse multi-channel transformer, and, potentially, an entropy decoder and other modules, to reconstruct the audio data from a quantized form.

[**0049**] The rate/quality controller (**170**) processes the information to determine a desired quantization step size given current conditions and outputs the quantization step size to the quantizer (**150**). The rate/quality controller (**170**) then measures the quality of a block of reconstructed audio data as quantized with the quantization step size, as described below. Using the measured quality as well as bitrate information, the rate/quality controller (**170**) adjusts the quantization step size with the goal of satisfying bitrate and quality constraints, both instantaneous and long-term. In alternative embodiments, the rate/quality controller (**170**) applies works with different or additional information, or applies different techniques to regulate quality and bitrate.

[**0050**] In conjunction with the rate/quality controller (**170**), the encoder (**100**) can apply noise substitution, band truncation, and/or multi-channel rematrixing to a block of audio data. At low and mid-bitrates, the audio encoder (**100**) can use noise substitution to convey information in certain bands. In band truncation, if the measured quality for a block indicates poor quality, the encoder (**100**) can completely eliminate the coefficients in certain (usually higher frequency) bands to improve the overall quality in the remaining bands. In multi-channel rematrixing, for low bitrate, multi-channel audio data in jointly coded channels, the encoder (**100**) can suppress information in certain channels (e.g., the difference channel) to improve the quality of the remaining channel(s) (e.g., the sum channel).

[**0051**] The MUX (**180**) multiplexes the side information received from the other modules of the audio encoder (**100**) along with the entropy encoded data received from the entropy encoder (**160**). The MUX (**180**) outputs the information in WMA or in another format that an audio decoder recognizes.

[0052] The MUX (180) includes a virtual buffer that stores the bitstream (195) to be output by the encoder (100). The virtual buffer stores a pre-determined duration of audio information (e.g., 5 seconds for streaming audio) in order to smooth over short-term fluctuations in bitrate due to complexity changes in the audio. The virtual buffer then outputs data at a relatively constant bitrate. The current fullness of the buffer, the rate of change of fullness of the buffer, and other characteristics of the buffer can be used by the rate/quality controller (170) to regulate quality and bitrate.

[0053] With reference to FIG. 2, the generalized audio decoder (200) includes a bitstream demultiplexer ["DEMUX"] (210), an entropy decoder (220), an inverse quantizer (230), a noise generator (240), an inverse weighter (250), an inverse multi-channel transformer (260), and an inverse frequency transformer (270). The decoder (200) is often simpler than the encoder (100) because the decoder (200) does not include modules for rate/quality control.

[0054] The decoder (200) receives a bitstream (205) of compressed audio data in WMA or another format. The bitstream (205) includes entropy encoded data as well as side information from which the decoder (200) reconstructs audio samples (295). For audio data with multiple channels, the decoder (200) processes each channel independently, and can work with jointly coded channels before the inverse multi-channel transformer (260).

[0055] The DEMUX (210) parses information in the bitstream (205) and sends information to the modules of the decoder (200). The DEMUX (210) includes one or more buffers to compensate for short-term variations in bitrate due to fluctuations in complexity of the audio, network jitter, and/or other factors.

[0056] The entropy decoder (220) losslessly decompresses entropy codes received from the DEMUX (210), producing quantized frequency coefficient data. The entropy decoder (220) typically applies the inverse of the entropy encoding technique used in the encoder.

[0057] The inverse quantizer (230) receives a quantization step size from the DEMUX (210) and receives quantized frequency coefficient data from the entropy decoder (220). The inverse quantizer (230) applies the quantization step size to the quantized frequency coefficient data to partially reconstruct the frequency coefficient data. In alternative embodiments, the inverse quantizer applies the inverse of some other quantization technique used in the encoder.

[0058] The noise generator (240) receives from the DEMUX (210) indication of which bands in a block of data are noise substituted as well as any parameters for the form of the noise. The noise generator (240) generates the patterns for the indicated bands, and passes the information to the inverse weighter (250).

[0059] The inverse weighter (250) receives the weighting factors from the DEMUX (210), patterns for any noise-substituted bands from the noise generator (240), and the partially reconstructed frequency coefficient data from the inverse quantizer (230). As necessary, the inverse weighter (250) decompresses the weighting factors. The inverse weighter (250) applies the weighting factors to the partially reconstructed frequency coefficient data for bands that have not been noise substituted. The inverse weighter (250) then adds in the noise patterns received from the noise generator (240).

[0060] The inverse multi-channel transformer (260) receives the reconstructed frequency coefficient data from the inverse weighter (250) and channel mode information from the DEMUX (210). If multi-channel data is in independently coded channels, the inverse multi-channel transformer (260) passes the channels through. If multi-channel data is in jointly coded channels, the inverse multi-channel transformer (260) converts the data into independently coded channels. If desired, the decoder (200) can measure the quality of the reconstructed frequency coefficient data at this point.

[0061] The inverse frequency transformer (270) receives the frequency coefficient data output by the multi-channel transformer (260) as well as side information such as block sizes from the DEMUX (210). The inverse frequency transformer (270) applies the inverse time-split transform selectively (as indicated by the side information), and applies the inverse of the frequency transform used in the encoder and outputs blocks of reconstructed audio samples (295).

Exemplary Transform with Selective Time-Split

[0062] FIG. 3 shows a transform coder 300 with selective time-split transform. The transform coder 300 can be realized within the generalized audio encoder 100 described above. The transform coder 300 alternatively can be realized in audio encoders that include fewer or additional encoding processes than the described, generalized audio encoder 100. Also, the transform coder 300 can be realized in encoders of signals other than audio.

[0063] A transform coder 110 need not employ adaptive window sizing. In one such an example, a default window size is used to transform coefficients from the time domain to the transform domain (e.g., frequency domain). Changing signal characteristic detection is used to determine where to selectively apply a time-split transform to coefficients in the frequency domain.

[0064] Optionally, a time-split transform may be used in conjunction with adaptive window sizing. The transform coder 300 utilizes a one or more pass process to select window sizes for transform coding. In a first, open-loop pass, the transform coder detects changing signal characteristics in the input signal, and selectively performs a time-split transform. An initial window configuration may or may not take changing signal characteristic detection into consideration. Optionally, window sizes may be adapted before or after selectively applying a time-split transform.

[0065] When window size adaptation is employed for an initial window-size configuration, the transform coder places one or more small windows over changing signal characteristic regions and places large windows in frames without changing signal characteristics. The transform coder first transform codes, time-split transforms (selectively) and then reconstructs the signal using the initial window configuration, so that it can then analyze auditory quality of transform coding using the initial window configuration. Based on the quality measurement, the transform coder adjusts window sizes, either combining to form larger windows to improve coding efficiency to achieve a desired bit-rate, or dividing to form smaller windows to avoid pre-echo. To save on computation, the transform coder 300 can use the quality measured on the previous frame to make

adjustments to the window configuration of the current frame, thereby merging the functionality of the two passes, without having to re-code.

[0066] With reference to a particular example shown in FIG. **3**, the transform coder **300** comprises components for changing signal characteristic detection **320**, windows configuration **330**, encoding **335**, and selective time-split transform **340**. Optionally **345**, quality measurement **350** is used to provide one or more window configurations **365**.

[0067] The changing signal characteristic detection component **320** detects regions of the input signal that exhibit characteristics of a changing signal characteristic, and identifies such regions to the windows configuration component **330**. The changing signal characteristic detection component **320** can use various conventional techniques to detect changing signal characteristic regions in the input signal. An exemplary changing signal characteristic detection process **400** is illustrated in FIG. **4**, and described below.

[0068] The windows configuration component **330** configures windows sizes for transform coding. An initial window configuration may be provided based on results of changing signal characteristic detection. An initial window configuration may also be provided by a default configuration without considering changing signal characteristic detection. An initial configuration may be determined on an open-loop basis based on the changing signal characteristic locations identified by the changing signal characteristic detector component **320**. An exemplary open-loop windows configuration process **500** is illustrated in FIG. **5**, and described below. Optionally, on a second iteration **365**, the windows configuration component **330** adjusts the initial window sizes from the initial configuration based on closed-loop feedback **365** from the quality measurement component **350**, to produce a next configuration. An exemplary closed-loop windows configuration process **700** is illustrated in FIG. **7**, and described below.

[0069] The encoding component **335** implements processes for transform coding (e.g., DCT transform, etc.), rate control, quantization and their inverse processes, and may encompass the various components that implement these processes in the generalized audio encoder **100** and decoder **200** described above. The encoding component **335** initially transform codes (with rate control and quantization) the input signal using the initial window size configuration produced by the windows configuration component **330**. The time-split component **340** then selectively performs a time-split transform, as described below. Optionally, when a decoder is employing feedback **365**, the encoding component **335** then decodes to provide a reconstructed signal for auditory quality analysis by the quality measurement component **350**. The encoding component **335** again transform codes (with rate-control and quantization) the input signal using the second-pass window size configuration provided by the windows configuration component **330** to produce the compressed stream **360**.

[0070] The quality measurement component **350** analyzes the auditory quality of the reconstructed signal produced from transform coding using the initial or next window size configuration, so as to provide closed-loop quality measurement feedback to the windows configuration component **330**. The quality measurement component analyzes the quality of each coding window, such as by measuring the noise-to-excitation ratio achieved for the coding window. Alternatively, various other quality measures (e.g., the noise-to-mask ratio) can be used to assess the quality achieved with the selected window size. Optionally, this quality measure is used by the windows configuration component **330** in its second-pass to select particular window sizes to increase for rate control, with minimal loss of quality.

[0071] The quality measurement component **350** may also use the quality analysis to detect pre-echo. An exemplary process to detect pre-echo is illustrated in FIG. **8**), and described below. Results of the pre-echo detection also are fed back to the windows configuration component **330**. Based on the pre-echo detection feedback, the windows configuration component **330** may further reduce window sizes (e.g., where rate-control constraints allow) to avoid pre-echo for the second-pass window configuration.

[0072] In the case of multi-channel audio encoding, the transform coder **300** in one implementation produces a common window size configuration for the multiple coding channels. In an alternative implementation for multi-channel audio encoding, the transform coder **300** separately configures transform window sizes for individual coding channels.

Exemplary Changing signal characteristic Detection

[0073] FIG. **4** illustrates one exemplary changing signal characteristic detection process **400** performed by the changing signal characteristic detection component **320** to detect changing signal characteristics in the input signal. As indicated at step **470**, the process **470** is repeated on a frame-by-frame basis on the input signal.

[0074] The changing signal characteristic detection process **400** first band-pass filters (at first stage **410**) the input signal frame. The changing signal characteristic detection process **400** uses three filters with pass bands in different audio ranges, i.e., low, middle and high-pass ranges. The filters may be elliptic filters, such as may be designed using a standard filter design tool (e.g., MATLAB), although other filter shapes alternatively can be used. The squared output of the filters represents the power of the input signal in the respective audio spectrum range at each sample. The low-pass, mid-pass and high-pass power outputs are denoted herein as $P_l(n)$, $P_m(n)$, and $P_h(n)$, where n is the sample number within the frame.

[0075] Next (at stage **420**), the changing signal characteristic detection process **400** further low-pass filters (i.e., smoothes) the power outputs of the band-pass filter stage for each sample. The changing signal characteristic detection process **400** performs low-pass filtering by computing the following sums (denoted $Q_l(n)$, $Q_m(n)$ and $Q_h(n)$) of the low-pass, mid-pass and high-pass filtered power outputs at each sample n, as shown in the following equations:

$$S_l(n) = \sum_{i=0}^{t} P_l(n-s+i)$$

$$S_m(n) = \sum_{i=0}^{t} P_m(n-s+i)$$

-continued

$$S_h(n) = \sum_{i=0}^{t} P_h(n - s + i)$$

where s and t are predefined constants and (t≧s). Examples of suitable values for the constants are t=288 and s=256.

[0076] The changing signal characteristic detection process **400** then (at stage **430**) calculates the local power at each sample by again summing the power outputs of the three bands over a smaller interval centered at each sample, as shown by the following equations:

$$Q_l(n) = \sum_{i=0}^{v} P_l(n - u + i)$$

$$Q_m(n) = \sum_{i=0}^{v} P_m(n - u + i)$$

$$Q_h(n) = \sum_{i=0}^{v} P_h(n - u + i)$$

where u and v are predefined constants smaller than t and s. Examples of suitable values of the constants are u=32 and v=32.

[0077] At stage **440**, the changing signal characteristic detection process **400** compares the local power at each sample to the low-pass filter power output, by calculating the ratios shown in the following equations:

$$R_l(n)=S_l(n)/Q_l(n)$$

$$R_m(n)=S_m(n)/Q_m(n)$$

$$R_h(n)=S_h(n)/Q_h(n)$$

[0078] Finally, at decision stage **450** and **460**, the changing signal characteristic detection process **400** determines that a changing signal characteristic exists if the ratio calculated at stage **440** exceeds predetermined thresholds, T_l, T_m, and T_h for the respective bands. In other words, if any of $R_l(n)>T_l$, or $1/R_l(n)>T'_l$, or $R_m(n)>T_m$, or $1/R_m(n)>T'_m$, or $R_h(n)>T_h$, or $1/R_h(n)>T'_h$, where T, T'_l, T_m, T'_m, T_h, T'_h are thresholds, then the sample location n is marked as a changing signal characteristic location. An example of suitable threshold values is in the range of 10 to 40. It is important to note that a changing signal characteristic is declared so long as there is sufficient change in energy in any of the three bands. So coding efficiency may be reduced if there are certain frequency ranges where a changing signal characteristic did not exist.

Exemplary Window Configuration

[0079] FIG. **5** shows an open-loop window configuration process **500**, which is used in the window configuration component **530** to perform its first pass window configuration. Adaptive window size configuration is not required to perform time-split transforms in a transform coder, rather it is an additional feature that may be employed in some embodiments. The open-loop window configuration process **500** configures window sizes for transform coding by the encoding component **340** based on information of changing signal characteristic locations detected via the changing signal characteristic detection process **400** by the changing signal characteristic detection component **320**. In the illustrated process, the window configuration component **330** selects from a number of predefined sizes, which may include a smallest size, largest size, and one or more intermediate sizes.

[0080] As indicated at step **510** in the window configuration process **500**, the process **500** determines if any changing signal characteristics (CSC), such as a transient or otherwise were detected in the frame. If so, the window configuration process places windows of the smallest size over changing signal characteristic-containing regions of the frame (as indicated at **520**), such that the changing signal characteristics are completely encompassed by one or more smallest size windows. Then (at **530**), the process **500** fills gaps before and after the smallest size windows with one or more transition windows.

[0081] If no changing signal characteristics are detected in a frame, the window configuration process **500** configures the frame to contain a largest size window (as indicated at **540**). The process **500** continues on a frame-by-frame basis as indicated at step **550**.

[0082] FIG. **6** shows an example window configuration produced via the process **500**. First, since no changing signal characteristic is detected in the prior frame, the process **500** places a largest size window **610** in that frame. The process **500** then places smallest size windows **620** to completely encompass changing signal characteristics detected in a transient region. The process **500** next fills a gap between the window **610** and windows **620** with intermediate size transition windows **630** and **640**, and also fills a gap with the next frame window with intermediate size transition window **650**. The open-loop window configuration process **500** has the advantage that the smallest size windows are placed over the changing signal characteristic region, as compared to filling a full frame.

Exemplary Quality Measurement

[0083] As discussed above, an optional quality measurement component **350** analyzes the achieved quality of audio information and feeds back the quality measurements to the window configuration component for use in adjusting window sizes. A window configuration component **350** may take two actions depending on the achieved quality of the signal. First, when the quantization noise is not acceptable, the window configuration component **350** trades the time resolution for better quantization by increasing the smallest window size. Further, when pre-echo is detected, the window configuration component splits the corresponding windows to increase time resolution, provided there are sufficient spare bits to meet bit rate constraints.

[0084] More specifically, FIGS. **7** and **8** show a quality measurement and adapted window configuration process **700**. As indicated at decisions **710** and **810**, a bit rate setting can be considered in the transform coder **300** (FIG. **3**) in order to determine whether the process **700** takes the actions depicted for processing loops **720-750** and **820-840**, respectively. More particularly, when a bit rate setting emphasizes coding efficiency (at **710**), the window configuration process **700** performs processing loop **720-750**. When the rate

setting is for high quality (at **810**), the window configuration process **700** performs processing in loop **820-840**. These rate setting classes need not be mutually exclusive. In other words, there may be some rate settings in some transform coders that call for a balance of both coding efficiency and quality, such that both processing loops **720-750** and **820-840** are performed.

[**0085**] At a first processing step **720** in the first processing loop **720-750**, the window configuration process **700** measures the achieved quality of the transform coded signal. In one implementation, the process **700** measures the achieved Noise-To-Excitation Ratio (NER) for each coding window. The NER of the coding window of the reconstructed, transform coded signal can be calculated as described in the Perceptual Audio Quality Measurement Patent Application, U.S. patent application Ser. No. 10/017,861, filed Dec. 14, 2001. Alternatively, other quality measures applicable to assessing acceptability or perceptibility of quantization noise can be used, such as noise-to-mask ration described or referenced in "Method for objective measurements of perceived audio quality," International Telecommunication Union-Recommendation Broadcasting Service (Sound) Series (ITU-R BS) 1387 (1998).

[**0086**] Next (at **730**), the window configuration process **700** compares the quality measurement to a threshold. If the quantization noise is not acceptable, the window configuration process **700** (at **750**) increases the minimum allowed window size for the frame. As an example, in one implementation, the window configuration process **700** increases the minimally allowed window size for the frame by a factor of 2 if the NER of a coding window in the frame exceeds 0.5. If the NER is greater than 1.0, the minimum allowed window size is increased by 4 times. The acceptable quantization noise threshold and the increase in minimum allowed window size are parameters that can be varied in alternative implementations.

[**0087**] As indicated at decision **740**, the window configuration process **700** also can increase the window size when the quantization noise is acceptable, but the rate control buffer of the transform coder is nearly full (e.g., 95% or other like amount depending on size of buffer, variance in bit rate, and other factors).

[**0088**] In an alternative implementation of the process **700**, the window configuration process **700** at processing step **720** uses a delayed quality measurement. As examples, the quality of coding of the preceding frame or average quality of previous few frames could be used to determine the minimum allowed window size for the current frame. In one implementation, the final NER obtained at the preceding frame is used to determine the minimum window size (at **750**) used in the configuration process **500**. Such use of a delayed quality measurement reduces the implementation complexity, albeit with some sacrifice in accuracy.

[**0089**] In the second processing loop **820-840**, the window configuration process **700** also measures to detect pre-echo in the frame. For pre-echo detection, the process **700** divides the frame of the reconstructed, transform coded signal into a set of very small windows (smaller than the smallest coding window), and calculates the quality measure (e.g., the NMR or NER) for each of the very small windows. This produces a quality measure vector (e.g., a vector of NMR or NER values). The process **700** also calculates a global achieved quality measure for the frame (e.g., the NMR or NER of the frame). The process **700** determines that pre-echo exists if any component of the vector is significantly higher (e.g., by a threshold factor) than the achieved global quality measure for the frame. Suitable threshold factor is in the range 4 to 10. Alternative implementations can use other values for the threshold.

[**0090**] In the case where pre-echo is detected and there is sufficient spare coding capacity (e.g., rate control buffer not full or nearly full), the window configuration process **700** (at **840**) adjusts the window configuration in the frame to further reduce the window size. In one implementation, the process **700** decomposes the frame into a series of smallest size windows (e.g., the size of window **620** of FIG. **6**). Alternatively, the process **700** locally reduces the size of the first-pass coding windows in which pre-echo is detected, rather than reducing all windows in the frame to the smallest size. As indicated at **850**, the window configuration process **700** then continues on a frame-by-frame basis. However, alternative implementations need not perform the window configuration on a frame basis.

Exemplary Selective Time-Splitting

[**0091**] In order to determine whether to apply time-splitting, the data is programmatically examined for certain characteristics (see e.g., FIG. **3**, **320**). In another example (not shown), after encoding (**335**), the results are examined for pre or post echo or other artifacts, such as changing signal characteristics (**320**, **350**). Pre-echo or post-echo are common characteristics of using a large time window when a small one is needed.

[**0092**] Optionally, an input signal is coded into a baseband and then the baseband shape is examined to determine similar shapes in an extended band. A similar shape in the baseband provides a shape model for similar shapes being coded at other frequencies. The baseband shapes provide synthetic models or codewords used to code the higher frequencies. The coded baseband is used to create an extended band or enhanced layer. In one such example, a time envelope is created resulting from reconstructing with the enhancement layer and comparing with the original time envelope. If there is a big difference, in the original versus reconstructed signal, then a determination is made to time-split at or near sub-bands where signal quality is compromised between the enhanced and original signal.

[**0093**] A changing signal characteristic detection routine (**320**) should also look for large energy differences in a high band which is being coded in the enhancement layer. If there are significant energy differences only present in the high band (such as, those being coded with enhancement in the extended band), and not in frequencies which are being coded with the baseband codec, then this is the ideal case when a large window size should be used for the baseband. Then, time-splitting can be used for the enhancement to get better time resolution in high frequencies without requiring a shorter window in the baseband. This will give the best compression efficiency without causing undesirable artifacts due to poor time resolution at high frequencies.

[**0094**] However, there might be cases when artifacts remain even after performing the time-split transform. Although the time-split results in energy compaction in time domain, it does not always work as well as truly using a

smaller time window (e.g., see smaller windows in FIG. **9**, **908**). In such a case, the results from time-split can be used as feedback before deciding to modify the window size. This means that if the high band is not able to be coded well (e.g., acceptable artifacts), then simply reduce the sub-frame size being used (e.g., FIG. **3**, **365**).

[**0095**] Additionally, it will be apparent that any similarly suitable and invertible transform can be used to alter or dampen the artifacts created by spreading the error across the spectrum. Here, since the MLT is an orthogonal transform, applying a orthogonal transform keeps the overall transform still orthogonal. The effect it has is in modifying the basis functions.

Exemplary Overlapping Windows

[**0096**] When utilizing an MLT (e.g., MDCT), overlapping windows are used to segment the data into blocks. For each of these overlapping blocks, a DCT transform is performed on the data in the window. Optionally, plural overlapping window sizes can be used. The windows sizes can be applied based upon signal characteristics, where small windows are used at changing signal characteristics (e.g., where signal characteristics such as energy change), and larger windows are used elsewhere to obtain better compression efficiency.

[**0097**] FIG. **9** is a graph representing exemplary overlapping windows covering segmentation blocks. As shown in **900**, the segmentation blocks **902** of signal data are transformed from the time domain to the transform domain (e.g., frequency domain) using overlapping windows **904**. For each window with M spectral samples, an overlapping window of size 2M (50% overlap on each side) is used to transform the data. However, the coefficients in the 2M window may not all be nonzero coefficients, as this depends on the neighboring block sizes. If either of the two neighboring blocks and corresponding windows are smaller than M, then at least some of the 2M window coefficients are zero.

[**0098**] For each block (or sub-frame), an invertible transform is computed transforming input audio samples from the time domain to the transform domain (e.g., a DCT or other known transform domains). The M resulting MLT coefficients from the 2M window are used for each M-size sub-frame. The overlap ensures that this 2M-to-M transformation can be inverted without any loss. Of course, there will be some loss during quantization. The 2M-to-M transformation can be represented as a projection of the 2M-dimensional signal vector onto the basis vectors. The shape of the M basis vectors are dependent on the window shape. Neither overlapping windows nor any particular segmentation methodology is required to time-split adjacent coefficients. However, if overlapping windows are used, the basis vectors typically vary based on the current sub-frame size, the previous sub-frame size, and the next sub-frame size. If the DCT cosine basis vectors (e.g., basis vectors) are to provide good time resolution, then they should have localized support in the time domain. If the basis vectors are viewed as a function of time index, then they should have most of their energy concentrated around the center of the frame.

Exemplary Segmentation

[**0099**] Consider an example, with a 32-dimensional vector (e.g., 32 input samples, such as audio/video), that has been split into 4 sub-frames of size 8 (e.g., a segmentation of [8 8 8 8]). Often, the frame would be larger (e.g., 2048 samples) and the segmentation (e.g., sub-frame sizes) would be larger and possibly variable in size within the frame (e.g., [8, 8, 64, 64, 32, 128, 128]). As will be discussed, time-splitting transform can be selectively performed without regard to sub-frame size and whether or not segmentation size is variable. However, the 32-dimensional vector provides an example for the following discussion, with the understanding that the described technology is not limited to any such configurations.

[**0100**] FIG. **10** is a graph of the basis vectors that contribute to the MLT coefficients corresponding to the middle two sub-frames. For example, assume that the 16 basis vectors **1000**, each with time span **16**, contribute to the MLT coefficients from the middle two sub-frames (e.g., in bold [8 8 8 8]). This illustrates **1000** that each sub-frame has a certain time span, and the time resolution is related to the time span. Similarly, if the 32 dimensional vector is segmented into 8 sub-frames of size 4, then the segmentation would be [4 4 4 4 4 4 4 4].

[**0101**] FIG. **11** is a graph of the basis vectors that contribute to the MLT coefficients corresponding to the middle four sub-frames with smaller sized segmentation. For example, the basis vectors **1100** corresponding to the MLT coefficients for the middle **4** sub-frames (e.g., [4 4 4 4 4 4 4 4]), are the same differently grouped coefficients as the middle **2** sub-bands in the sub-band size 8 case. The basis vectors **1100** each have a time-span of 8. The graph **1100**, shows the basis vectors as a time frequency grid, with the time axis running along the columns, and the frequency axis being the rows.

[**0102**] From these two FIGS. **1000**, **1100**, it is apparent that sub-frame size relates to time resolution. Now, suppose that the time resolution is sufficient at lower frequencies (e.g., **1002**), but not at higher frequencies (e.g., **1004**). Note that in FIG. **10**, the top row is the lowest frequency basis vector, and each row below it, in order, increases in frequency with the bottom row being the highest frequency. For example, if there is a changing signal characteristic in a high frequency, it may be beneficial to provide better time resolution to reduce artifacts introduced by the changing signal characteristic. However, it may only be the high frequency that has a changing signal characteristic, and thus the time resolution in a lower frequency is adequate **1002**. Also, the time resolution needed for a particular frequency range is also dependent on the coding method being used to code that frequency range. For example, when coding a particular frequency range as an extended band using "Efficient coding of digital media spectral data using wide-sense perceptual similarity", then better time resolution might be needed than if coding it as a traditional baseband coding scheme.

[**0103**] A time-splitting transform is selectively applied at adjacent coefficients where better time resolution is desired. Instead of just using the coefficients obtained from the MLT, a post block transform on a subset of the M spectral coefficients is performed, such as a time-splitting transform. By imposing constraints on the structure of the transform, better time resolution is selectively obtained for some frequency coefficients, but not others.

[**0104**] FIG. **12** is a graph representing how time-splitting combines adjacent coefficients. In this example, the com-

bined coefficients are high frequencies coefficients. As shown, the basis vectors **1-4** remain unchanged **1202**, but basis vectors **5**±6, and 7±8 have been selected for a time-split **1204**. Thus, basis vectors **5** and **6** have been added to and subtracted from one another to provide a time-split transform. Basis vectors **7** and **8** have been added to and subtracted from one another to provide a time-split transform. In this example, two sets of basis vectors have been transformed to represent time-splitting, but either could be used alone, such as just 5±6, or 7±8. Additionally, the 8 rows of adjacent basis vectors could provide various other selectable time-splitting transforms, such as one or more of the following row transforms: 1±2, 2±3, 3±4, 4±5, 5±6, 6±7, or 7±8. Thus, any basis vector can be time-split with any adjacent basis vector. The graph **1200** represents how a 5±6, 5−6 and 7±8, 7−8 time-split transform relates to the basis vectors. A selective application of time-splitting is applied to the high frequency coefficients, for example, using a simple transform of the form (a+b)/2, (a−b)/2, where 'a' and 'b' are two adjacent coefficients. Notice that FIG. **11** provides rows of four frequency patterns and columns of four (shifting) time patterns. Further, FIG. **10** provides rows of eight frequency patterns and columns of two time patterns. In one respect, time splitting as shown in FIG. **12** provides better time resolution of FIG. **11** for a sample selection of high frequencies, while maintaining the better frequency resolution for low frequencies of FIG. **10**.

[0105] FIG. **13** is a matrix representing an exemplary time-splitting transform of FIG. **12**. The time-splitting transform represented by FIG. **12**, is applied after the time domain to frequency domain (e.g., DCT) transform, in this example using the matrix **1300**. By combining (±) basis functions from different frequencies, frequency resolution is reduced, and time resolution is gained in the process. Better time resolution is useful to more closely model rapidly changing data from a transient area. For example, using the time-split transform on the example of sub-frame size 8, the high frequency basis functions from FIG. **11**, are effectively incorporated into the basis vectors shown in FIG. **12**. The $1/\sqrt{2}$ scaling factor can be optionally applied, as shown in FIG. **13**, to maintain proper normalization of the time-split basis functions (such as those in FIG. **12**, **1204**). Alternatively, that normalization factor can be incorporated in the quantization steps of the encoding component **335**. Also, other values for the normalization factor can be used, if it is deemed appropriate, e.g. by the quality measurement **350**.

[0106] As can be seen, the post block transform (e.g., time-split transform) results in time separation. Although the time span of the resulting basis vectors is the same as before, the energy concentration has been more localized. This is better understood in view of the following analysis.

Exemplary Analysis of Time-splitting

[0107] The MLT coefficients for a sub-frame of size M are defined as:

$$X[k]=\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1}x[n]h[n]\cos\left[\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{M}\right],\quad k=0,1,\ldots,M-1, \qquad \text{Equation 1}$$

where h[n] is the window. The time index n=0 is defined to be M/2 samples to the left of the start of the current sub-frame, so that x[M/2] is the start of the current sub-frame. Notice that the equation provides an optional overlapping window sizes (e.g., 2M). Starting with X[k]+X[k+1], and then using the known relationship of cos(a)+cos(b)=2 cos((a−b)/2)cos((a+b)/2), the following is obtained:

$$X[k]+X[k+1]=2\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1}x[n]h[n]\cos\left[\left(n+\frac{M+1}{2}\right)(k+1)\frac{\pi}{M}\right]\cos\left[\left(n+\frac{M+1}{2}\right)\frac{\pi}{2M}\right] \qquad \text{Equation 2}$$

Similarly, staring with X[k]-X[k+1], and using the known relationship of cos(a)-cos(b)=−2 sin((a−b)/2)sin((a+b)/2), the following is obtained:

$$X[k]-X[k+1]=2\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1}x[n]h[n]\sin\left[\left(n+\frac{M+1}{2}\right)\frac{\pi}{2M}\right]\sin\left[\left(n+\frac{M+1}{2}\right)(k+1)\frac{\pi}{M}\right] \qquad \text{Equation 3}$$

Equations 2 and 3 can be rewritten as equations 4 and 5, respectively, as follows,

$$X[k]+X[k+1]=2\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1}x[n]h_1[n]\cos\left[\left(n+\frac{M+1}{2}\right)(k+1)\frac{\pi}{M}\right] \qquad \text{Equation 4}$$

$$X[k]+X[k+1]=2\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1}x[n]h_2[n]\sin\left[\left(n+\frac{M+1}{2}\right)(k+1)\frac{\pi}{M}\right] \qquad \text{Equation 5}$$

such that $h_1[n]$ and $h_2[n]$ are defined as shown in equations 7 and 8.

$$h_1[n]=h[n]\cos\left[\left(n+\frac{M+1}{2}\right)\frac{\pi}{2M}\right]$$

$$h_2[n]=h[n]\sin\left[\left(n+\frac{M+1}{2}\right)\frac{\pi}{2M}\right] \qquad \text{Equation 7 and 8}$$

Thus, the two original frequency-domain coefficients X[k] and X[k+1], which corresponded to the modulating frequencies (k+1/2)π/M and (k+3/2)π/M, respectively, are replaced. By replacing those coefficients with the following coefficients (X[k]+X[k+1]) and (X[k]−X[k+1]), there are two new frequency-domain coefficients that now correspond to the same frequency (k+1)π/M (but with a 90 degree phase shift, since one is modulated by a cosine function and the other by a sine function), but modulated by different windows $h_1[n]$ and $h_2[n]$, respectively.

[0108] FIG. **14** is a graph of these two new time-split window functions. In this example, the graph is of the two new window functions of Equations 7 and 8 plotted with M=256. The graph of the two equations shows why the time separation occurs. Assuming the neighbor windows have the same window shape, the standard sub-frame window shape **906** used in FIG. **9**, is represented as follows,

$$h[n] = \sin\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2M}\right],\ n = 0, 1, \ldots, 2M-1. \qquad \text{Equation 9}$$

[0109] A sub-band merging approach was first described in R. Cox, "The Design of Uniformly and Nonuniformly Spaced Pseudoquadrature Mirror Filters" IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 34, pp. 1090-1096, October 1986, and was applied to the MLT in H. S. Malvar, "Enhancing the Performance of Sub-Band Audio Coders for Speech Signals", Proc. 1998 IEEE International Symposium on Circuits and Systems, vol. 5, pp. 98-101, June 1998 ("Malvar"). Contrary to the decomposition in Malvar, where all high-frequencies (after a predetermined value of k) are pairwise split according to the construction above, a time-splitting transform is performed only on selected pairs of coefficients, according to a selection criteria. Thus, fixed time-splitting is replaced by selective time-splitting based upon the characteristics of the input signal or derivations thereof. In practice one can combine more then two sub-bands, but the quality of time-splitting suffers that is, time selectivity will not be as good. See e.g., O. A. Niamut and R. Heusdens, "Sub-band Merging in Cosine-modulated Filter Banks", IEEE Signal Processing Letters, vol. 10, pp. 111-114, April 2003, and see ADAPTIVE WINDOW-SIZE SELECTION IN TRANSFORM CODING, U.S. patent application Ser. No. 10/020,708 filed Dec. 14, 2001.

Exemplary Spectral Coefficients

[0110] FIG. **15** is a graph representing a set of spectral coefficients. For example, the coefficients (**1500**) are an output of a sub-band transform or an overlapped orthogonal transform such as MDCT or MLT, to produce a set of spectral coefficients for each input block of the audio signal.

[0111] In one example, a portion of the output of the transform called the baseband. (**1502**) is encoded by the baseband coder. Then the extended band (**1504**) is divided into sub-bands of homogeneous or varied sizes (**1506**). Shapes in the baseband (**1508**) (e.g., shapes as represented by a series of coefficients) are compared to shapes in the extended band (**1510**), and an offset (**1512**) representing a similar shape in the baseband is used to encode a shape (e.g., sub-band) in the extended band so that fewer bits need to be encoded and sent to the decoder.

[0112] Sub-bands may vary from subframe to subframe. Similarly, a baseband (**1502**) size may vary, and a resulting extended band (**1504**) may vary in size based on the baseband. The extended band may be divided into various and multiple size sub-band sizes (**1506**).

[0113] In this example, a baseband segment is used to identify a codeword for a particular shape (**1508**) to simulate a sub-band in the extended band (**1510**) transformed to create other shapes (e.g., other series of coefficients) that might more closely provide a model for the vector (**1510**) being coded. Thus, plural segments in the baseband are used as potential models to code data in the extended band. Instead of sending the actual coefficients (**1510**) in a sub-band in the extended band an identifier such as a motion vector offset (**1512**), is sent to the encoder to represent the data for the extended band. However, sometimes there are no close matches in the baseband for data being modeled in a sub-band. This may be because of low bitrate constraints that allow a limited size baseband. The baseband size (**1502**) as relative to the extended band may vary based on computing resources such as time, output device, or bandwidth.

Exemplary Transform Matrices

[0114] One channel of audio/video is split into time segments as shown in FIG. **9**, and for each segment a time domain to frequency domain transformation is provided, optionally with an overlapping windows. For example, assume a overlapping window is applied to a time segment followed by a DCT. A DCT produces coefficients which are linear projections of the windowed time segment onto basis vectors. The inverse frequency to time domain transformation involves taking a linear combination of the basis vectors where the basis vectors are weighted by the DCT coefficients. Thus any noise (e.g. quantization noise) or other significant energy changes in the DCT coefficients will be spread across time due to the support of the basis vectors. For example, if the basis vectors have compact support (e.g. the energy is localized in time), then there will be less temporal smearing of the quantization noise or other changes to the DCT coefficients. One way to do this is to break a time window into smaller windows. Since the basis vectors have a support of 2M samples, the smaller the M, the smaller the support. Another way is to selectively use a time-split transform in frequency ranges where changing signal characteristics are detected, or in frequency ranges where it is needed because of the coding method being used. A time splitting transform does not reduce the support of the basis vectors, but instead compacts the energy into different regions of the time segment. Therefore there will be some energy over the entire 2M samples of the basis vector, but a large portion of it will be concentrated around a central point.

[0115] FIG. **16** is a graph of an exemplary time domain representation of frequency coefficients. For example, if a signal at a frequency changes dramatically (**1604**), preferably a window size that is adequate for a more stable signal (**1602**) should be divided into smaller windows to reduce echo. But in the context of frequency extrapolation using a baseband and extended band codec, not all windows can be sub-divided. In one example, a baseband window represented frequency information coding up to 10-kilohertz (kHz) (**1606**), and under 10 kHz, there is generally no need to break windows up because the sound is quite uniform. However, above 10 kHz, for example to 20 kHz, there might be a distinct sound such as a metallic sound that would show up in the 20 kHz frequency range. For this distinct sound in a determined frequency, a time-split is possibly performed to provide better time resolution. Thus, one or more frequencies within a larger segment are selectively time split. A larger window is used for the base transform but a time split achieves better time resolution for selected frequencies within the larger window.

[0116] As shown in FIG. **16**, a time domain may be divided into Low, Medium or High Frequency (e.g., L, M, H, etc). Other resolutions may be used for examining inputs for data variance requiring time-split or window adaptation. It can be any of the bands H, L, M, that need better time resolution. Frequency coefficients (**1608**) are represented in the time domain as x[n], for n=0 . . . 2M-1. The idea is to identify any segment that could beneficially use better time resolution and then apply a transformation that is going to add and subtract two coefficients together to alter the basis vector support. Any linear transform can be described as a matrix multiplication; however, they are often implemented in a more efficient way (e.g., Fast Fourier Transform).

[0117] FIG. **17** is a diagram representing a linear transformation of a time domain vector into a frequency domain vector including a time-split transform matrix of FIG. **13**. For example, a vector from the time domain (**1608**, **1702**) is multiplied by a cosine basis matrix (**1704**) and a time split matrix (**1706**) to create a transform domain vector (**1708**) (e.g., frequency domain vector). The matrix **1704** contains the coefficients of the operator corresponding to the cascade combination of the signal-domain window and a DCT (of type IV), such as the MLT. Thus, the number of coefficients in the signal x[n] in the time domain is 2M, and the number of transform-domain (or frequency-domain) coefficients X[k] is M, indexed from 0 to M-1. Each element in the cosine matrix is given by Equation 9 above except for selected frequencies, which are represented selectively in the time-split-matrix by Equations 7 and 8 above.

[0118] Thus, at the decoder, when the frequency domain coefficients X[k] are transformed back to the time domain, each vector **1708** is multiplied by similar orthogonal matrices. The selected frequencies within the basis vectors **1704**, are effectively multiplied by the basis vectors show in FIG. **14**, thereby a changing signal characteristic in the input signal due to the larger window, is not spread throughout the selected frequencies because the time-split reduces the energy achieving a reduced or zero value. This modulated cosine is shifted a little bit in frequency, and creates a shape that reduces an error such as an echo. In this example, this result is achieved by multiplying by a second time-split transform matrix **1706**, that effectively combine two adjacent coefficients.

[0119] As shown in FIG. **13**, at whatever frequency region time-split is desirable, a 2×2 block is inserted (**1302**) into the time-split matrix. For example, two adjacent basis vectors can be combined **1302**, **1304**, as shown in FIG. **13**. However in practice, combining more than two sets has not been effective.

[0120] The time-split transform should be done prior to quantization, but after the first transform **1704**. For example, a time split transform **1706** could also be applied before or after the channel transform **120**, but before quantization **150** and before the weighter **140**. A 2×2 block can be place along the diagonal selectivity (as shown in FIG. **13**) in order to obtain better time resolution. The transform could also be placed in a 3×3 block, 4×4 block, but the results have not proven as successful as a 2×2 block. Additionally, 2×2 blocks can be placed in various positions and the results of each position is compared upon reconstruction to determine a best placement. For example, the blocks can be transformed one way, then other ways, and the best results are selected for final coding. In another example, frequency regions for time-split transform are dynamically selected for frequency regions or for multiple frequency regions via some form of energy change detection. The results are compared, and for each eligible 2×2 block position, a bit is set to indicate whether the time-split transform is on or off. Intuitively, a transform is more likely to apply to high energy blocks since they often spread more energy.

[0121] A time-split transform is a selectively applied sum and difference of adjacent coefficients. For example, a time-split transform may also be called a selectively applied sum-difference of adjacent coefficient orthogonal transform (e.g., a SASDACO transform). Additionally, the coder signals the decoder in an output stream, where to orthogonally apply the inverse transform. For example, a side-information bit for each frequency pair signals where to apply the time-splitting SASDACO transform, and eligible blocks may be anywhere along the diagonal (e.g., two examples in FIG. **13**, **1302**, **1304**) or only in the enhanced frequency (**1504**).

[0122] Of course, a sum-difference orthogonal transform 2×2 block is not limited to the 2×2 block shown in FIG. **13**, **1302**. For example, a transform coder could utilize any orthogonal sum-difference transform with similar transformational properties. In one such example, a orthogonal sum-difference transform on adjacent coefficients results in transforming a vector of coefficients in the transform domain as if they were multiplied by an identity matrix with at least one 2×2 block along a diagonal of the matrix, where the at least one 2×2 block comprises orthogonally transformational properties substantially similar to one of the following 2×2 blocks:

$$c*\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$c*\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

where c is a scale factor selected to vary the properties of the transform.

[0123] In one example, an extended portion **1504** of a sub-frame is signaled (e.g., a bit) as with or without time-split. A signaled sub-frame, may further signal a sub-band **706** as time-split, and signal blocks to perform a SASDACO transform. In one such example, a signaled block implicitly indicates applying a SASDACO transform to the other sub-bands in the sub-frame. In another example, a signal(s) is provided for each sub-band **706**. A pre-echo/post decisions can be used to decide where to apply the time-split transform. A changing signal characteristic detection component may also be used to break a signal up into frequency ranges, such as high, medium, and low. For these distinctions, the transform coder determines whether there is a change in energy and applies a SASDACO transform accordingly.

Exemplary Additional Features

[0124] Thus, a block transform (e.g., time-split transform) is used after MLT decomposition to selectively get better

time resolution for only some frequency components. This is useful when larger time windows can be used to get better compression efficiency, for example with low, medium, or frequency coefficients, and still provide better time resolution only where needed. A decision is used to select where to perform time-split, by programmatically examining characteristics of the spectral data. For example, examining a time envelope, energy change, changing signal characteristic detection, pre-echo, or post-echo. A decision where to perform time-split may instead be made by programmatically examining characteristics of changing signal characteristic detection. In another example, modification (reduction) of sub-frame size for base coding is made by programmatically examining the output of enhancement layer coding. These various ways of making a decision of where to make a time-split transform, may also be used to determine in a second pass at coding, where to vary window size.

Exemplary Computing Environment

[0125] FIG. **18** illustrates a generalized example of a suitable computing environment (**1800**) in which the illustrative embodiment may be implemented. The computing environment (**1800**) is not intended to suggest any limitation as to scope of use or functionality of the invention, as the present invention may be implemented in diverse general-purpose or special-purpose computing environments.

[0126] With reference to FIG. **18**, the computing environment (**1800**) includes at least one processing unit (**1810**) and memory (**1820**). In FIG. **18**, this most basic configuration (**1830**) is included within a dashed line. The processing unit (**1810**) executes computer-executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory (**1820**) may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory (**1820**) stores software (**1880**) implementing an audio encoder.

[0127] A computing environment may have additional features. For example, the computing environment (**1800**) includes storage (**1840**), one or more input devices (**1850**), one or more output devices (**1860**), and one or more communication connections (**1870**). An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment (**1800**). Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment (**1800**), and coordinates activities of the components of the computing environment (**1800**).

[0128] The storage (**1840**) may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, CD-RWs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment (**1800**). The storage (**1840**) stores instructions for the software (**1880**) implementing the audio encoder.

[0129] The input device(s) (**1850**) may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing environment (**1800**). For audio, the input device(s) (**1850**) may be a sound card or similar device that accepts audio input in analog or digital form. The output device(s) (**1860**) may be a display, printer, speaker, or another device that provides output from the computing environment (**1800**).

[0130] The communication connection(s) (**1870**) enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, compressed audio or video information, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

[0131] The invention can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment (**1800**), computer-readable media include memory (**1820**), storage (**1840**), communication media, and combinations of any of the above.

[0132] The invention can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

[0133] For the sake of presentation, the detailed description uses terms like "determine,""get,""adjust," and "apply" to describe computer operations in a computing environment. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

[0134] Having described and illustrated the principles of our invention with reference to an illustrative embodiment, it will be recognized that the illustrative embodiment can be modified in arrangement and detail without departing from such principles. It should be understood that the programs, processes, or methods described herein are not related or limited to any particular type of computing environment, unless indicated otherwise. Various types of general purpose or specialized computing environments may be used with or perform operations in accordance with the teachings described herein. Elements of the illustrative embodiment shown in software may be implemented in hardware and vice versa.

[0135] In view of the many possible embodiments to which the principles of our invention may be applied, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.

We claim:

1. A transform coder comprising:

a changing signal characteristic detection component operating to identify a changing signal characteristic location;

an encoding component transforms an input signal from a time domain to a transform domain; and

a time-splitting transformer component operating in response to the identified changing signal characteristic location to selectively perform an orthogonal sum-difference transform on adjacent coefficients indicated by the identified changing signal characteristic location.

2. The transform coder of claim 1 wherein the orthogonal sum-difference transform on adjacent coefficients results in transforming a vector of coefficients in the transform domain as if they were multiplied by an identity matrix with at least one 2×2 block along a diagonal of the matrix, where the at least one 2×2 block comprises orthogonally transformational properties substantially similar to one of the following 2×2 blocks:

$$c*\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$c*\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

where c is a scale factor.

3. The transform of claim 1 wherein the time-splitting transformer component encodes side information indicating that there are no time-splits in at least one sub-frame.

4. The transform coder of claim 1 further comprising:

a window configuration component operating in response to the identified changing signal characteristic location to configure a first configuration of window sizes selected from at least a small window size and a large window size, so as to place one or more windows of the small window size to encompass a region of the input signal having at least one identified changing signal characteristic location and place windows of the large window size in areas of the input signal having no identified changing signal characteristic locations;

the encoding component further inverse-transforming to produce a reconstructed version of the input signal;

a quality measurement component operating to measure achieved quality of the reconstructed signal; and

the window configuration component operating in response to the achieved quality measurement to adjust sizes of the first configuration of window sizes according to the achieved quality measurement to produce a second configuration window sizes.

5. The transform coder of claim 4 wherein:

the quality measurement component further operates to measure achieved perceptual quantization noise of the reconstructed signal for at least some of the windows in the first configuration; and

the window configuration component further operates to increase a window size where the measure of achieved perceptual quantization noise exceeds an acceptable threshold.

6. The transform coder of claim 4 wherein:

the quality measurement component further operates to detect pre-echo in the reconstructed signal; and

the window configuration component further operates to decrease at least one window where pre-echo is detected.

7. The transform coder of claim 2 wherein the transform coder outputs side information identifying where the orthogonal sum-difference transform was performed.

8. A transform decoder that performs the corresponding inverse steps to recover data coded by the transform coder of claim 7.

9. A transform decoder comprising:

an inverse time-splitting transformer component receives side information and coefficient data in a transform domain and selectively performs an inverse orthogonal sum-difference transformation on adjacent coefficients indicated in received side information; and

after the inverse time-splitting transformer component performs an inverse orthogonal sum-difference transformation, an inverse transformer transforms coefficient data from the transform domain to a time domain.

10. The transform decoder of claim 9 wherein the transform decoder outputs reconstructed audio samples.

11. The transform decoder of claim 9 further comprising:

an inverse window configuration component receives side information about window and sub-frame sizes; and

the inverse transformer transforms coefficient data according to the window and sub-band sizes.

12. The transform decoder of claim 9 wherein the inverse orthogonal sum-difference transformation on adjacent coefficients results in transforming a vector of coefficients in the transform domain as if it were multiplied by an inverse of a time-splitting transform used to code the vector of coefficients.

13. The transform decoder of claim 9 wherein the inverse time-splitting transformer component receives side information indicating that an inverse selectively applied sum-difference of adjacent coefficients orthogonal transform should be applied to at least one pair of adjacent coefficients in a vector X in the transform domain, where there are M coefficients in vector X that are uniquely identified as X[k] with k an integer ranging from 0 to M−1, so that the pair of adjacent coefficients is of the form {X[2r], X[2r+1]}, where r is an integer.

14. The transform decoder of claim 9 wherein the inverse time-splitting transformer component receives side information indicating that there are no time-splits in at least one sub-frame.

15. The transform decoder of claim 9 wherein the inverse time-splitting transformer component receives side information indicating whether or not there is a time-split in an extended band.

16. A method of decoding comprising:

receiving side information and coefficient data in a transform domain;

selectively performing an inverse time-split transform on adjacent coefficients as indicated in received side information; and

performing an inverse transform on received coefficient data comprising transforming the coefficient data from the transform domain to a time domain.

17. The method of claim 16 further comprising:

identifying sub-frame sizes in received side information; and

wherein the inverse transform is performed according to the identified sub-frame sizes.

18. The method of claim 16 wherein selectively performing the inverse time-split transform comprises determining from the side information whether there is a time-split in a sub-band.

19. The method of claim 16 wherein selectively performing the inverse time-split transform comprises determining from side information whether or not there is a time-split in each sub-band in an extended band.

20. The method of claim 16 wherein selectively performing the inverse time-split transform comprises determining from side information a pair of adjacent coefficients in a transform domain on which to perform an inverse sum-difference transform.

* * * * *